SIMULACIÓN DE ALTAS PRESTACIONES PARA MODELOS ORIENTADOS AL INDIVIDUO

Departamento de Arquitectura de Computadores y Sistemas Operativos

Memoria presentada por **Diego Javier Mostaccio** para optar al grado de **Doctor en Informática** por la Universidad Autónoma de Barcelona

Barcelona, Marzo de 2007

SIMULACIÓN DE ALTAS PRESTACIONES PARA MODELOS ORIENTADOS AL INDIVIDUO

Memoria presentada por Diego Javier Mostaccio para optar al grado de Doctor en Informática por la Universidad Autónoma de Barcelona. Trabajo realizado en el Departamento de Arquitectura de Computadores y Sistemas Operativos (DACSO) de la Escuela Técnica Superior de Ingeniería de la Universidad Autónoma de Barcelona, dentro del programa de Doctorado en Informática, opción A:"Arquitectura de Computadores y Procesamiento Paralelo", bajo la dirección del Dr. Remo Suppi Boldrito.

Director:

Dr. Remo Suppi Boldrito

Barcelona, Marzo de 2007

Agradecimientos

Los finales de los ciclos son momentos que sirven para hacer un alto, mirar atrás y darse cuenta de las cosas que han sucedido en todo ese tiempo. En este caso, este ciclo es una tesis doctoral en la cual hubo muchas personas que fueron fundamentales para que pueda estar escribiendo estas líneas y a las que les estaré siempre agradecido.

Cuando salí de Argentina traje muchas cosas (gran parte de ellas al día de hoy no las usé, típico en mí) pero, de todas ellas, la que más me sirvió y me seguirá sirviendo, es todo lo que mis padres me han enseñado. A ellos principalmente, les quiero dar las gracias por todas las "clases" que me dieron y me siguen dando y por mostrarme con su ejemplo que "la única lucha que se pierde es la que se abandona" en todos los aspectos de la vida. A mis hermanas, sobrinas y a Oscar también les hago extensivo mi agradecimiento por estar siempre cerca y apoyarme.

Un agradecimiento muy especial a Carol, mi compañera y gran amiga, por su apoyo, paciencia y cariño que han sido de sumo valor para afrontar la recta final de esta aventura.

Durante estos años tuve la suerte de conocer mucha gente y hacer nuevos amigos los cuales han sido muy importantes para poder afrontar este desafío. En estos agradecimientos trataré de no olvidar a nadie pero si así sucede, espero ya haberles transmitido a todos ellos mi satisfacción de contar con su amistad.

A mi director de tesis Remo Suppi, no solo le agradezco el tiempo dedicado al trabajo sino también al apoyo que me brindó en lo personal en los momentos "complicados".

A Emilio Luque por haberme dado la posibilidad de participar en el grupo de investigación y por haber confiado en mí. También quiero agradecerle las largas horas de discusión y la buena predisposición para ayudar en los temas de investigación que hicieron posibles conseguir los objetivos pensados.

A Lola por el apoyo y el optimismo que han sido tan importante en el "sprint".

A Miguel Angel Mayosky por su confianza, consejos y la ayudar para abrir las puertas de esta experiencia única.

A Eduardo Argollo por haberme “desobedecido” y por sobre todas las cosas, la gran amistad que tenemos.

A mis “primeros” compañeros de piso y de doctorado Germán y Mauricio con los que compartimos la llegada y el descubrimiento de Barcelona y que siempre me dieron su amistad y compañerismo.

A mi amiga Paula que ha sabido estar siempre brindándome su oído y su apoyo.

A los amigos brasileros Chris, Angelo, Leandro y Genaro (el “gurú”), por las charlas, los buenos momentos que pasamos juntos y las ideas que aportaron a este trabajo.

A Xiao, “mi vecino de escritorio” del que tanto aprendí en todos los aspectos y que fue mi “F1” tantas veces.

A Dani Ruiz, Jordi Valls y a la “libreta” por su excelente predisposición e idoneidad para la solución de todos los problemas que acontecieron con el Cluster.

A todos los compañeros de doctorado y a todos los integrantes de DACSO que generan un excelente ambiente de trabajo.

Índice de contenidos

Introducción *1*

1.1. Ciencia computacional **2**

1.2. Trabajos realizados y contribuciones a la simulación distribuida de IoM **4**

1.3. Organización de la tesis **10**

Simulación *11*

2.1. Introducción **11**

2.2. Simulación **13**

Modelos y tipos de sistemas simulados 14

Simulación dirigida por eventos y tiempo 16

2.3. Aceleración en la simulación **19**

Simulación paralela y distribuida 19

Simulación paralela vs distribuida 20

Niveles de Paralelismo/Distribución 20

2.4. Causalidad **22**

2.5. Paralelización de la simulación dirigida por eventos **23**

Método de simulación conservativo. 25

Método de simulación optimista 28

Modelado de Sistemas Biológicos *33*

3.1. Introducción **33**

3.2. Modelos biológicos **34**

3.3. Tipos de modelos biológicos **38**

Modelo orientado al Individuo (IoM). 38

Modelo orientado a la población, ecuaciones de Lotka-Volterra 39

3.4. Fish Schools **41**

Modelo biológico 43

Modelo matemático 52

Modelo computacional 58

Simulador *67*

4.1. Introducción **67**

4.2. Implementación del simulador **68**

4.3. Funcionamiento y algoritmo de simulación **69**

Detalles de la implementación 72

Simulador, detalles de las etapas 74

Sincronización para el método conservativo 79

4.4. Modelo analítico y de prestaciones del simulador **80**

Lenguaje de programación y librería de comunicación utilizados 87

4.5. Alternativa al modelo de simulación clásico **88**

Experimentación *99*

5.1. Introducción 99
5.2. Modelo biológico Fish School 99
5.3. Modelo analítico y prestaciones del simulador 102
5.4. Conclusión de las experimentaciones realizadas para demostrar la viabilidad de la propuesta 117
Conclusiones y Líneas abiertas 119
6.1. Conclusiones 119
6.2. Líneas abiertas 122
Bibliografía 125

Índice de figuras

Figura 1: Componentes de la ciencia computacional 4
Figura 2: Distribución del espacio de simulación 6
Figura 3: Metodología usada para el desarrollo de la herramienta 7
Figura 4: Mejoras en el rendimiento 8
Figura 5: Ejemplo de planificación entre eventos 16
Figura 6: Secuencia de eventos en la simulación 18
Figura 7: Ejemplo de paralelización 23
Figura 8: Arquitectura de la simulación de procesos lógicos 25
Figura 9: Arquitectura de un proceso lógico conservador 26
Figura 10: Interbloqueo y agotamiento de memoria 28
Figura 11: Arquitectura de un proceso lógico optimista 29
Figura 12: Áreas de influencia 44
Figura 13: Orientación paralela 46
Figura 14: Repulsión 47
Figura 15: Atracción 48
Figura 16: Distribución de probabilidad para el cálculo de β_{ij} 49
Figura 17: Típica distribución de frecuencia de velocidad en un banco de peces determinado experimentalmente y fijada por una distribución Gamma 50
Figura 18: Ejemplo de interacción con dos vecinos 51
Figura 19: Distribución de probabilidad en la interacción con dos individuos en la zona de orientación paralela 51
Figura 20: Ejemplo de reacción de repulsión 55
Figura 21: Discretización del espacio 60
Figura 22: Efecto de la discretización espacial 60
Figura 23: Caso 1, velocidades iguales e individuos no colineales 63
Figura 24: Arquitectura del simulador distribuido 68
Figura 25: Distribución del espacio de simulación en procesos lógicos 69
Figura 26: Individuo cerca de la frontera 71
Figura 27: Máquina de estados 72
Figura 28: Distintas etapas del simulador 73
Figura 29: Diagrama de flujo del simulador 75
Figura 30: Uso del producto escalar para la selección de los vecinos 77
Figura 31: Caso particular de varios vecinos con el mismo ángulo de respecto a la visión directa del i-ésimo pez 78
Figura 32: Distribución del espacio y regiones de envío de preguntas 81
Figura 33: Posición de un banco de peces en a) inicio de la simulación y en b) en la iteración *i-ésima* iteración 88
Figura 34: Reincorporación de individuos cuando se hallan fuera del mundo simulado 89
Figura 35: Seguimiento de un banco de peces 90
Figura 36: Distribución del espacio con varios bancos de peces 91
Figura 37: Desbalanceo de carga 91
Figura 38: Varios bancos de peces en un mismo mundo de simulación 92
Figura 39: Distribución de los bancos de peces en los procesadores 93
Figura 40: División del espacio de simulación en módulos 93

Figura 41: Módulo de Cómputo de cinco procesadores 94
Figura 42: Ejemplo de migración utilizando el *módulo de cómputo* 95
Figura 43: Unión de dos bancos de peces 100
Figura 44: Unión de dos bancos de peces sin aleatoriedad 101
Figura 45: Tiempo por iteración [Segundos] 106
Figura 46: Caso particular de ancho de los procesos lógicos 106
Figura 47: Tiempo por iteración para más de 64 procesadores 107
Figura 48: Tiempos por Iteración para el escenario E1 109
Figura 49: Tiempos por Iteración para el escenario E1 para el rango de 8-32 procesadores 109
Figura 50: Detalle de los tiempos utilizados 110
Figura 51: SpeedUp 111
Figura 52: Tiempo real y estimado [Segundos] 112
Figura 53: Tiempo real y estimado de 8 a 64 procesadores [Segundos] 113
Figura 54: Tiempo real y estimado para más de 64 procesadores [Segundos] 113
Figura 55: Escalabilidad en la complejidad del problema 115
Figura 56: Tiempos preestablecidos y estimados 115
Figura 57: Densidad 116
Figura 58: Error absoluto 116

Índice de tablas

Tabla 1: Predicciones con el modelo analítico del simulador 9
Tabla 2: Escenarios de Simulación 104
Tabla 3: Tiempos de simulación [Segundos] 105
Tabla 4: Cantidad de individuos y tamaño de los procesos lógicos 105
Tabla 5: Tiempos para el escenario E1 [Segundos] 108
Tabla 6: Escalabilidad en la complejidad del problema [Número de Individuos] 114
Tabla 7: Tiempos de simulación serie y distribuida [Seg/Iteración] 120

Capítulo 1

Introducción

El análisis y estudio del comportamiento de grupos de individuos es de interés para diversas ramas del mundo de la investigación que comprende desde biólogos hasta economistas. Esta conducta global puede ser el resultado de un modelo en el cual el conjunto de los individuos es considerado como un "todo" o bien puede ser que el comportamiento del grupo sea el resultado de la interacción entre los elementos que forman el sistema. En este último caso lo que se modela es la interacción entre individuos.

La simulación de la segunda clase de modelos para sistemas con cantidades considerables de elementos requiere de una elevada potencia de cómputo. En las primeras épocas en las que los computadores comenzaron a ser utilizados como elemento de cómputo para la obtención de mayores prestaciones (más velocidad) estaba vinculada con los avances tecnológicos a nivel de la arquitectura de los procesadores. La aparición de computadores paralelos permitió el desarrollo de nuevos planteamientos en la forma de implementar los simuladores. Uno de ellos es el concepto de "dividir y conquistar" en el que un problema es fraccionado en problemas menores o de menor complejidad (sean o no independientes) de forma tal que cada uno puede ser tratado al mismo tiempo por un elemento de cómputo distinto.

Los progresos en las redes de interconexión han dado paso a los "clusters" que son básicamente un conjunto de nodos de cómputo (pueden ser ordenadores personales) interconectados por una red de alta velocidad que dependiendo del uso que se les dé pueden trabajar como un computador paralelo [16] [68] [67]. Con esta tecnología se pueden obtener las altas prestaciones necesarias para la simulación de modelos que se basan en la interacción entre los individuos del sistema (Individual oriented Model, IoM) [21].

La implementación de un IoM en un simulador requiere de la interacción de diversas disciplinas. El sistema a simular es modelado primeramente por científicos del campo en el que se encuentra. La tecnología informática, como se ha mencionado anteriormente, es necesaria no sólo para brindar una plataforma de altas prestaciones sino también para la implementación del modelo. La obtención del IoM específico y del simulador son los dos extremos del proceso. En el medio se encuentra la matemática aplicada que es la encargada de representar el IoM matemáticamente para que, posteriormente, la informática lo transforme en un modelo computacional y pueda ser simulado.

La aplicación de estos campos del conocimiento para la resolución de un problema y la complejidad del mismo, determinan que la simulación de IoM con grandes cantidades de elementos se encuentre en el marco de la ciencia computacional [73].

1.1. Ciencia computacional

En la bibliografía pueden hallarse diversas definiciones de ciencia computacional:

- "Es la aplicación de técnicas computacionales y numéricas para resolver problemas complejos y grandes" [79].
- "Uso de computadores para estudiar problemas científicos y complementar las áreas teóricas y experimentales en la investigación científica tradicional" [78].
- "Enfoque interdisciplicario para la solución de problemas complejos que utilizan conceptos y técnicas provenientes de disciplinas de la ciencia, matemática e informática" [79].

De las definiciones mostradas anteriormente se deriva que la ciencia computacional permite afrontar problemas de grandes magnitudes haciendo uso de las ciencias experimentales, la matemática y la informática.

En la actualidad la ciencia computacional es considerada como el tercer paradigma para los descubrimientos científicos, aplicada en la resolución de problemas con base en la ciencia y la tecnología, más allá de la teoría y la experimentación. Estos dos últimos fueron considerados por muchos años las alternativas de trabajo en los campos del conocimiento científico.

La ciencia computacional no debe ser confundida con la Informática (Computer Science). La primera computacional se enfoca en problemas científicos o de ingeniería, y junto con la matemática y la informática obtiene un entendimiento perfeccionado de los problemas. Por otra parte, la informática centra su interés en el computador mismo (estudio matemático del cómputo, computadores y el procesamiento de la información). Aunque las áreas son bastante distintas, muchos de los temas típicamente considerados pertenecientes al dominio de "computer science" son de mucho valor en la ciencia computacional.

El término de "computational scientist" ha sido "acuñado" para describir científicos, ingenieros y matemáticos que aplican computadores de altas prestaciones para innovar y avanzar en el estado del conocimiento en sus respectivas disciplinas. Para dichas tareas, una herramienta fundamental es la simulación la cual es aceptada como la tercera metodología en la investigación científica, complementando las aproximaciones de la teoría y los experimentos. De esta forma, surge como un método poderoso e indispensable para analizar una diversidad de problemas en investigación, desarrollo de productos y procesos y fabricación. Ésta provee una comprensión cualitativa y cuantitativa de muchos fenómenos que son muy complejos para ser tratados con métodos analíticos, cuyos experimentos resultan muy caros o incluso peligrosos.

En la actualidad, muchos experimentos e investigaciones que tradicionalmente se han realizado en laboratorios, túneles de viento, etc son extendidos y/o remplazados por la simulación. Algunos estudios, como la integridad de un repositorio nuclear y los cambios climáticos globales, involucran escalas de tiempo que imposibilita la realización de experimentos físicos reales. La disponibilidad de computadores de altas prestaciones asociado con el avance en los algoritmos, posibilitan grandes adelantos en las investigaciones. No obstante aunque muchas veces no pueden reemplazar el laboratorio, tienen un lugar muy importante en la investigación para el conocimiento científico [13].

La ciencia computacional no toma solamente ventaja de las mejoras en la arquitectura de los computadores, sino también de los adelantos en algoritmos y técnicas matemáticas. A su vez,

permite afrontar problemas que anteriormente fueron muy difíciles de realizar debido a la complejidad de la matemática, el gran número de cálculos involucrados, o la combinación de ambos.

Las ideas anteriores dan una visión global del significado de la relación entre las ciencias expuestas y pueden ser resumidas en el gráfico de la figura 1. En él se observa que la ciencia computacional es una aproximación interdisciplinaria para la solución de problemas complejos.

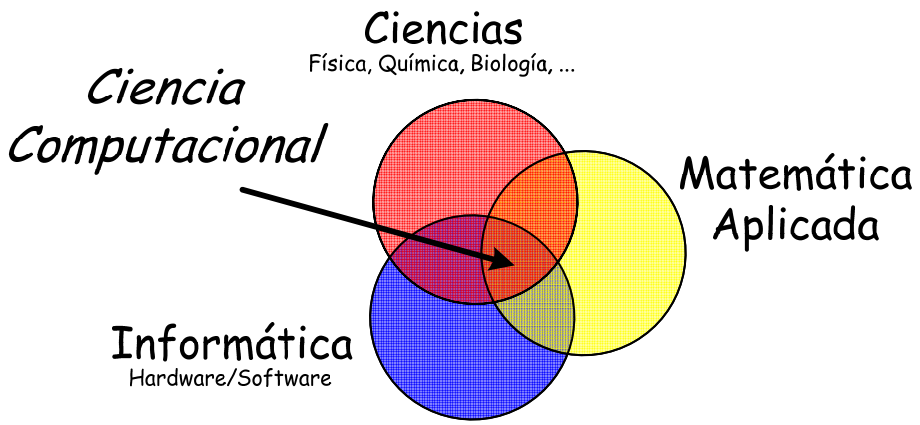


Figura 1: Componentes de la ciencia computacional

1.2. Trabajos realizados y contribuciones a la simulación distribuida de IoM

A lo largo de los últimos años se han realizado diversos trabajos en el grupo de investigación en temas vinculados a la simulación de sistemas biológicos. Una de las principales motivaciones ha sido la necesidad de biólogos y ecólogos de realizar estudios y análisis sobre el comportamiento de ciertas especies. Otro aspecto en el que se ha trabajado es en la simulación de altas prestaciones, entendiéndose a ésta como la que necesita de un alto poder de cálculo para llevar adelante la simulación.

La combinación de los dos campos mencionados anteriormente, son los motivadores de la línea de investigación en el área de la ciencia computacional. Estos se pueden ver reflejados en la simulación de sistemas biológicos con IoM aplicados a poblaciones con grandes cantidades de individuos como las que se encuentran en la naturaleza [48] [53] [72]. El modelo utilizado fue el Fish School desarrollado por Huth y Wiessel [37], el cuál representa el comportamiento individual de los peces. Mediante este modelo se puede

determinar el movimiento de un banco de peces como el resultado de la interacción entre los individuos del grupo.

El primer paso en el desarrollo del trabajo con modelos IoM fue la implementación de un simulador serie [72]. El modelo implementado era una simplificación del Fish Schools. Las simulaciones para poblaciones pequeñas empleaban tiempos razonables, es decir, eran lo suficientemente rápidas como para proporcionar resultados a un entorno gráfico para la visualización en tiempo real. Si se trabaja con poblaciones grandes, la frecuencia con la que se obtienen resultados es menor y por lo tanto la utilización de un entorno gráfico debe ser "post mortem". Con el objetivo de visualizar los datos de la simulación se desarrolló también una herramienta gráfica adecuada a las necesidades del simulador [71].

Como se ha comentado al principio del capítulo, la simulación de este tipo de modelos requiere de una importante potencia de cómputo. Esto es consecuencia de que para calcular la nueva posición de un individuo éste debe interactuar con el resto de los peces. De esta forma la complejidad del algoritmo es $O(N^2)$, con N cantidad de individuos a simular. Si el banco está constituido por una elevada cantidad de peces los tiempos de procesamiento se incrementan notablemente (en función de N^2). La disminución del tiempo fue uno de los objetivos del presente trabajo y se la consiguió por medio de la distribución del modelo de simulación aprovechando las características espaciales del mismo.

El espacio donde transcurre la simulación es finito, tiene forma de paralelepípedo y los peces se hallan distribuidos en forma aleatoria en todo el volumen. La distribución se hizo dividiendo el espacio en partes iguales y asignándole a cada procesador el cómputo de los peces que se encuentran dentro de cada uno de ellos (figura 2). Así, cada procesador calcula en paralelo el volumen asignado que tiene una complejidad de $O((N/P)^2)$ donde P es la cantidad de procesadores utilizados.

Una vez verificado que la distribución del simulador presenta considerables ganancias temporales respecto al simulador serie, se realizó una implementación rigurosa del modelo de Huth y Wiessel siguiendo la metodología propuesta por [46] para la resolución de problemas enmarcados en la ciencia computacional.

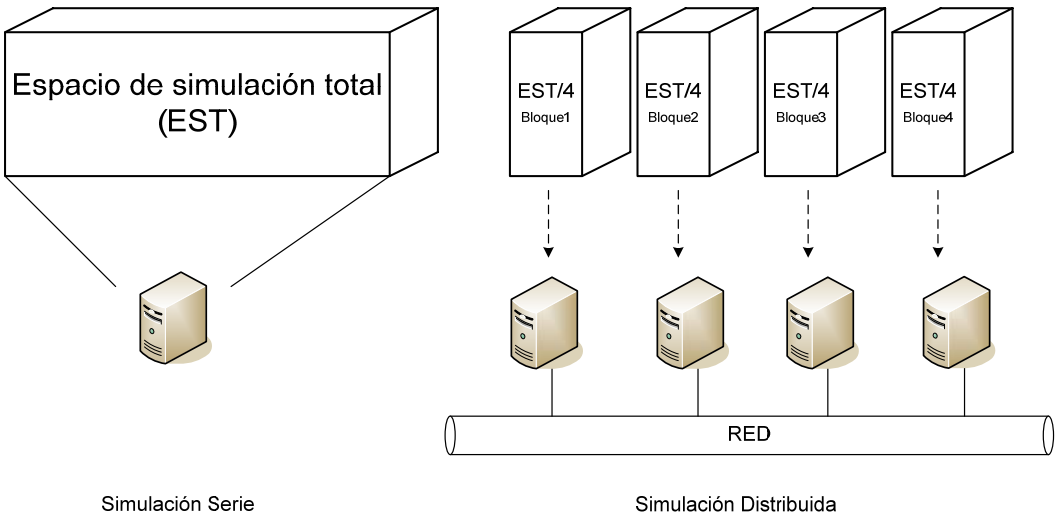


Figura 2: Distribución del espacio de simulación

Los pasos y etapas por las que se debieron transitar en el desarrollo del simulador son los que se encuentran unidos por las líneas discontinuas en la figura 3 [46]. Analizando el comportamiento del sistema físico se ha hecho un modelo que lo describe. En el caso del modelo IoM Fish Schools, este paso fue realizado por Huth y Wissel. Ellos, utilizando datos empíricos, transformaron la interacción entre individuos en un modelo analítico.

Si bien el modelo biológico o cualitativo se corresponde con la realidad, Huth y Wiessel desarrollaron un modelo bidimensional. Parte del trabajo que se ha realizado en esta tesis ha sido la extensión del modelo al espacio tridimensional. Para tal fin se han analizado con rigor el modelo biológico bidimensional para trasladarlo al espacio tridimensional.

Por lo tanto, el paso entre “Sistema Físico Actual” y “Modelo Analítico del Sistema”, que consiste en definir matemáticamente el modelo biológico, se ha tenido que replantear no solamente haciendo uso de la matemática, sino también del estudio del comportamiento de los peces y su interacción.

Como se puede apreciar en la figura 3, el paso marcado como “Análisis” necesita de dos de las componentes de la ciencia computacional, las ciencias, en este caso la biología y la matemática aplicada.

Una vez obtenido el modelo matemático se ha generado el modelo computacional, el cuál permite trasladar los comportamientos expresados matemáticamente a un simulador. Para este paso se han tenido en cuenta diversas cuestiones vinculadas con la plataforma donde se

ejecutaría el simulador (cluster) y cuestiones intrínsecas al paso de un modelo matemático a uno computacional como la discretización del tiempo y del espacio.

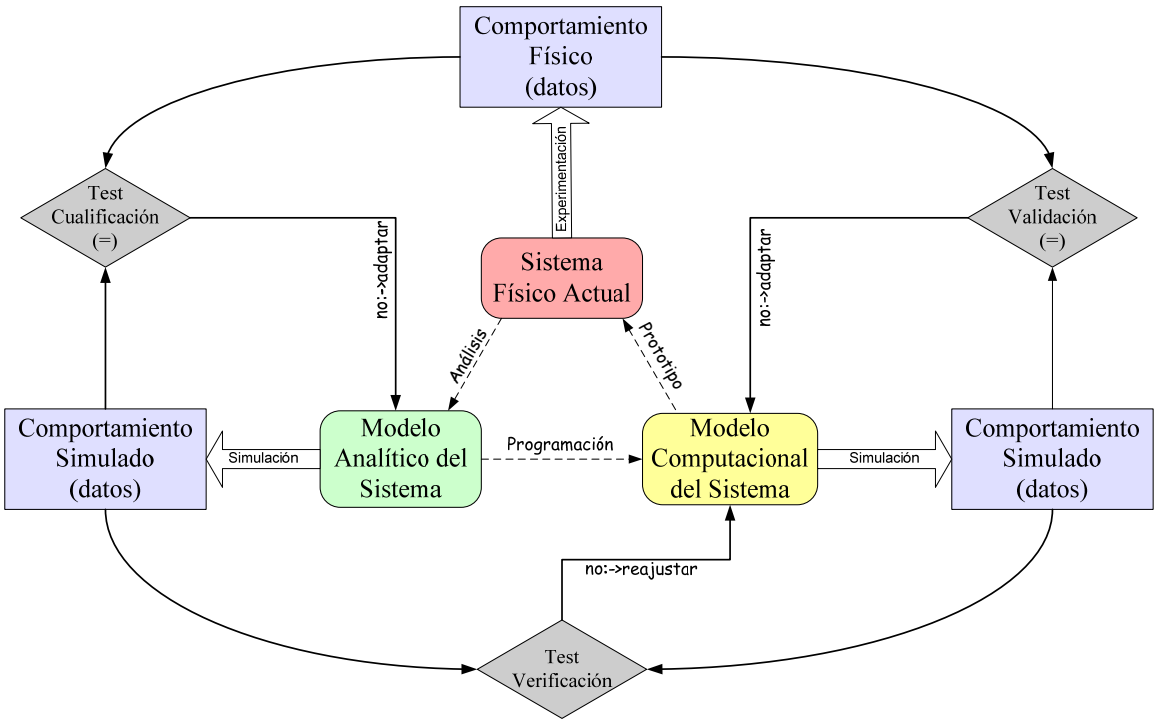


Figura 3: Metodología usada para el desarrollo de la herramienta

En este punto, el modelo computacional se verificó, es decir, se determinó que la implementación del modelo computacional representaba correctamente el modelo conceptual del sistema físico. En el caso de encontrar diferencias se debe reajustar el modelo computacional para obtener el funcionamiento correcto. El modelo conceptual o biológico fue verificado mediante una extensiva experimentación para determinar que las reglas de comportamiento modeladas eran cumplidas por el simulador.

Finalizada la etapa de verificación se pasó a la de validación en la que se determinó el grado en el cual el modelo computacional era una representación acertada del sistema físico desde la perspectiva de los usos previstos del modelo. Puesto que la finalidad del uso de esta herramienta es el estudio del comportamiento de los grupos utilizando a los individuos como elemento básico de simulación, la verificación se realizó comparando los resultados obtenidos de las simulaciones con datos experimentales obtenidos de la observación de bancos de peces reales (comportamiento físico).

Los últimos dos pasos, la programación y prototipo son realizadas por la tercera componente de la ciencia computacional que es la informática.

La descripción de la forma en la que se han realizado los diferentes modelos y cómo se han validado y verificado se puede tomar como base para la solución de problemas en los que interaccionan diversas disciplinas de la ciencia. El aporte de este trabajo no es solamente el desarrollo de una herramienta sino la metodología a seguir para enfrentar problemas que puedan enmarcarse en la ciencia computacional.

La figura 4 permite ver otra parte del trabajo realizado, la cual proporciona otro aporte a la simulación de altas prestaciones aplicada a IoM.

La parte izquierda de la figura 4 refleja la metodología explicada anteriormente mientras que en la derecha se halla el trabajo realizado en cuestión de rendimiento.

Una vez que se llega a una correcta validación del simulador, se plantea el tema de las prestaciones. Como se ha comentado al inicio del capítulo, las simulaciones de este tipo necesitan de la distribución del modelo para ser realizadas en tiempos reducidos. En el lazo de rendimiento de la figura 4 se observa nuevamente la acción de la informática pero ya no para la implementación del modelo sino para la mejora en el tiempo de simulación.

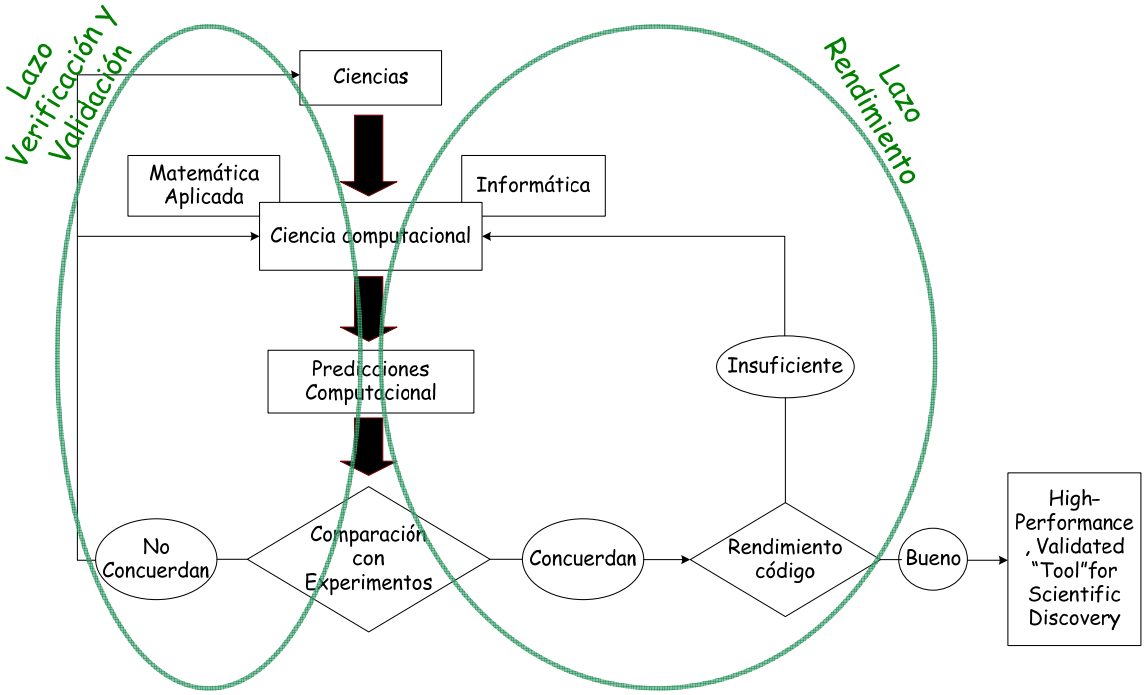


Figura 4: Mejoras en el rendimiento

Puesto que en esta etapa se hacen modificaciones en el código, se tiene que verificar el simulador después de cada modificación o mejora. La propuesta en el área informática es replantear la forma clásica de distribución del espacio en la que el espacio total de simulación

es dividido en tantas partes como procesadores se dispongan. Esta propuesta tiene como objetivo poder simular espacios tan grandes como sean necesarios (más reales) sin tener la limitación del volumen (por ejemplo el mostrado en la figura 2) y manteniendo el balanceo de carga.

Para poder afrontar simulaciones de espacios como los de la propuesta es necesario disponer de una potencia de cómputo considerable. El análisis de la escalabilidad del algoritmo y los límites de éste es otro de los temas desarrollado en el trabajo. Para tal fin se realizó un modelo analítico del simulador que permite predecir tres comportamientos (tabla 1). Con estos datos el usuario puede prever cuanto tiempo tardará una simulación de acuerdo a la cantidad de recursos (procesadores) disponibles y la cantidad de peces a simular. Esta predicción es de utilidad para analizar el comportamiento del simulador frente a la escalabilidad de procesadores, es decir, conocer como se modifica el tiempo de simulación con la variación de los recursos.

La cantidad de procesadores es la segunda de las predicciones y permite estimar la cantidad de recursos necesarios para una simulación en la que se desea utilizar un determinado tiempo y cantidad de peces.

La tercera de las estimaciones es la cantidad de peces que es posible simular para un tiempo de simulación y recursos específicos.

Tabla 1: Predicciones con el modelo analítico del simulador

Predicción	*Datos*
Tiempo de simulación	Cantidad de procesadores y de peces
Cantidad de procesadores	Tiempo de simulación y cantidad de peces
Cantidad de peces	Tiempo de simulación y cantidad de procesadores

Un simulador de estas características y el trabajo realizado en torno a él, brinda la posibilidad a los biólogos de disponer de una herramienta que les permite analizar el comportamiento de un banco de peces partiendo de un IoM, además de facilitar la estimación de los recursos necesarios para realizar la simulación deseada.

1.3. Organización de la tesis

En el siguiente capítulo se desarrollarán los conceptos básicos de la simulación que van desde la simulación serie hasta la distribuida. El modelo biológico, matemático y computacional son explicados con detenimiento en el capítulo 3 y en el siguiente cómo se ha realizado y cómo funciona el simulador. Además se describe la forma en la que se ha desarrollado el modelo analítico del simulador para predecir el comportamiento del mismo. En el mismo capítulo se hace una propuesta formal y detallada de la alternativa a la distribución del modelo espacial.

En el capitulo 5 se pueden observar las pruebas realizadas para la validación del modelo y las que muestran la viabilidad de la propuesta. Por último, en el capítulo 6, se encuentran las conclusiones y líneas abiertas obtenidas del trabajo realizado en la presente tesis.

Capítulo 2

Simulación

2.1. Introducción

El estudio o análisis de sistemas que existen o no físicamente puede llevarse a cabo de diversas maneras. En el caso de sistemas reales o que existen físicamente, una forma es observando las respuestas del sistema frente a determinados estímulos. Un inconveniente de esta metodología es el coste asociado cuando se pretende aplicar a cierto tipo de sistemas específicos. Un ejemplo de lo comentado anteriormente puede ser el caso de querer analizar el comportamiento de una pieza mecánica de difícil y elevado valor de construcción frente a variaciones de temperatura y cargas mecánicas. Como resultado de la experimentación se puede llegar al deterioro e inutilización del elemento en estudio.

Otro ejemplo de sistemas en los cuales no puede aplicarse aquella metodología sería el análisis de las consecuencias generadas en el ecosistema por fallas en un reactor nuclear o en una represa hidroeléctrica. En estos casos, más allá del coste, que en algunas situaciones puede hacer imposible la aplicación de esta metodología, existe el factor de la inviabilidad por tratarse de experimentos destructivos y/o peligrosos. Por tales motivos se puede apreciar que alguna otra técnica para el análisis de sistemas como éstos es necesaria.

Dicha técnica mediante la cual se puede duplicar o imitar el comportamiento de sistemas modelados mediante el uso de computadores es la simulación.

Anteriormente se han mencionado algunos ejemplos de sistemas reales o que existen físicamente y en los cuales no se puede realizar un análisis en forma empírica para determinar su comportamiento en situaciones extremas. Para estos sistemas la simulación es una herramienta de análisis de gran importancia.

Existen otros tipos de sistemas, los que serán implementados físicamente. En éstos, la simulación puede ser un mecanismo de ayuda para el diseño de ellos. De ésta forma el diseñador puede prever el comportamiento del sistema mediante la simulación. Un ejemplo de lo dicho anteriormente es el diseño y desarrollo de circuitos electrónicos.

La variedad de problemáticas que pueden ser abordadas mediante la técnica de simulación son muy diversas. Ejemplos de éstas son: control de tráfico automovilístico [10], logística en el transporte [74] o modelos que representan fenómenos naturales o a la misma naturaleza [12]. En este último grupo se encuentra el modelo con el que se ha trabajado en esta tesis el cual es un modelo biológico que permite el análisis y estudio del comportamiento de una especie.

Para la realización de la presente tesis, el sistema modelado se basa en la simulación del individuo como elemento básico del sistema para luego obtener el comportamiento final del sistema como la interacción entre los individuos (IoM) [35] [8] [64]. El modelo utilizado se denomina Fish School y modela la interacción entre peces. La complejidad en la simulación de este modelo hace que los tiempos necesarios para conseguir resultados sean elevados.

Con en el propósito del estudio de la simulación utilizando modelos que representen ecosistemas o partes de éstos como es el caso de Fish Schools, se han desarrollado distintos tipos de simuladores con diferentes métodos de simulación.

El primer paso en el estudio ha sido la simulación serie, desarrollándose para tal fin un simulador secuencial para ser ejecutado en un computador monoprocesador. Los tiempos resultantes de esta simulación fueron muy elevados para poblaciones grandes [72] [52].

La utilización de técnicas de paralelismo/distribución del modelo fue el siguiente paso. Para este fin se ha experimentado con distintos métodos de simulación distribuida como la conservativa y la optimista [54] [49] [51], obteniéndose mejoras considerables en los tiempos de simulación.

En este capitulo se hará un estudio de la simulación en general, haciendo un énfasis mayor en la simulación distribuida que es uno de los temas desarrollados en esta tesis.

2.2. Simulación

En la literatura se pueden encontrar diversas definiciones del término simulación. Entre todas ellas la que se ajusta más a los objetivos de esta tesis es la siguiente:

"*Sistema que permite duplicar o imitar el comportamiento de un sistema mas complejo mediante el uso de ordenadores. Los sistemas simulados pueden existir o no físicamente.*"

La utilización de simuladores brinda la posibilidad de analizar la evolución del sistema simulado frente a la variación de parámetros. De esta forma se pueden realizar conjeturas sobre el modelo que se pueden aplicar al sistema sin necesidad de manipulear, modificar o desarrollar el sistema real.

La simulación como herramienta de análisis consta de una serie de características que la tornan interesante para su utilización:

- Gran flexibilidad: se pueden modificar fácilmente diversos parámetros del sistema para su posterior estudio. A través de la simulación se pueden estudiar los efectos de ciertos cambios sobre la operación de un sistema, realizando alteraciones en el modelo y observando su efecto sobre el comportamiento del sistema modelado.

- Eficiencia: la simulación puede comprimir el tiempo de manera que largos intervalos sobre el sistema real transcurren en pocos segundos sobre el modelo simulado. Esto permite ahorrar gran cantidad tiempo durante el análisis del sistema real.

- Aislamiento con respecto del sistema físico: hace posible el estudio y experimentación con complejas iteraciones de un sistema dado o parte del mismo sin afectar al sistema real.

- Mejora de la comprensión del sistema real: una detallada observación del sistema que es simulado puede conducir a un mejor conocimiento del mismo y provocar mejoras sobre el sistema real que de otra forma no serían visibles. La simulación de sistemas complejos pueden brindar conocimientos de cuales son las variables más importantes del sistema y como se relacionan.

- Experimentación con sistemas físicos: la simulación puede ser utilizada para experimentar con nuevas situaciones sobre las cuales existe poca o ninguna información, verificando que ocurre con el sistema real. Cuando se introducen nuevos componentes en un sistema, la simulación puede ayudar a encontrar zonas de conflicto u otros problemas que comprometerían la operación del sistema real [18].

La simulación se aplica a un amplio espectro de disciplinas, por ejemplo:

- Ingeniería: simulación de circuitos, simulación de sistemas de microondas, simulación de sistemas mecánicos.
- Climatología: simulación de las variaciones climáticas [62] [31].
- Prevención de incendios: simulación del avance del frente de un incendio [2] [3] [25] [11].
- Comunicación: simulación de sistemas de video bajo demanda (VoD) [77] [76].
- Ecología: simulación de sistemas reales aplicando modelos orientados a la población o al individuo [72] [52] [54] [49] [51] [53] [50] [71].
- Gestión de recursos en cluster de ordenadores: [33] [34].

Modelos y tipos de sistemas simulados

Mediante la simulación "se imita" la ocurrencia de los eventos y como se produce el avance del sistema en el tiempo. La ocurrencia de los eventos implica en algunos casos la modificación de los estados del sistema.

Los eventos pueden estar planificados para que su ocurrencia siga algún tipo de regla o bien pueden generarse nuevos eventos a partir de la variación de los estados.

La simulación puede ser de dos tipos: continua o discreta. En una simulación continua, el cambio de estado ocurre en forma continua en el tiempo mientras que en una simulación discreta la ocurrencia de un evento es instantánea y fijada por un punto seleccionado en el tiempo [24].

Los sistemas a simular pueden ser clasificados dentro de alguno de los siguientes grupos [6]:

- Sistemas continuos.
- Sistemas discretos.

En los sistemas de la primera categoría se encuentran aquellos que modifican sus variables de estado en forma continua en el tiempo mientras que los sistemas del segundo grupo cambian el valor de sus variables de estado en un conjunto discreto de instantes en el tiempo. Un ejemplo del primer tipo de sistemas podría ser un circuito eléctrico analógico en el cual los elementos del sistema, como son las corrientes y las tensiones, varían en forma continua en el tiempo. Algunos ejemplos del segundo grupo podrían ser los circuitos digitales a nivel de compuertas o la transferencia entre registros, en los cuales su estado varía en instantes discretos en el tiempo.

Dependiendo de la naturaleza y los resultados que se deseen obtener puede modelarse un sistema de una clase mediante un modelo de otra clase. Por ejemplo, un sistema continuo podría ser simulado con un modelo discreto [6].

Los sistemas físicos o reales se modelan como un conjunto de procesos físicos (PF). La cantidad de PF que pueden formar parte del modelo del sistema puede variar entre uno o más según sean las características del sistema modelado. Cada PF representa alguna componente del sistema real simulado.

La evolución en el tiempo de un sistema implica la evolución de los PF vinculados a éste y para que el avance de estos últimos se efectúe correctamente es necesario un reloj global. La función de este reloj es la de marcar un único tiempo de referencia para el sistema completo.

Puesto que los PF son los elementos que forman el sistema modelado a simular operan en forma autónoma durante el transcurso de la simulación menos en los momentos en los que deben interactuar. Estas interacciones son denominadas eventos y pueden ser modelados por el intercambio de mensajes que contienen la siguiente información:

- Evento que representan.
- Timestamp (instante de tiempo en el que debe ser procesado el evento).

La estructura del mensaje está formada por el tiempo en el cual ocurrirá el mensaje (timestamp) y el tipo de mensaje o evento al cual pertenece. Este mensaje es intercambiado por los procesos físicos cuando deban comunicarse o intercambiarse información. Por ejemplo, una estructura posible sería:

mensaje <*e,t*>

Cuando un PF tiene previsto interactuar con otro, lo hace por medio del envío de mensajes como se ha explicado previamente. Los PFs receptores consumirán los mensajes que tengan el mismo tiempo de ocurrencia (timestamp) que el reloj global. El consumo de eventos puede generar el cambio en el estado del PF. Como consecuencia se pude dar el caso entre otros, de que se planifiquen nuevos eventos para el futuro y por lo tanto el envío de nuevos mensajes o cancelar eventos que estaban a la espera de que el tiempo indicado por el reloj sea igual a su tiempo de ocurrencia.

Un ejemplo de la aplicación de estos mensajes en un sistema compuesto por cuatro PFs y un reloj global se puede apreciar en la figura 5. En éste, el PF1 planifica interactuar con el PF3 en el tiempo 10. Entonces, el PF3 consumirá el evento modelado por el mensaje, cuando el reloj global tenga el valor 10.

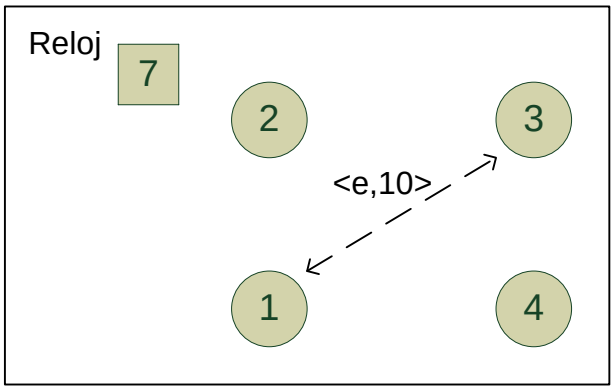


Figura 5: Ejemplo de planificación entre eventos

Simulación dirigida por eventos y tiempo

Hasta el momento se ha hecho referencia a los sistemas, los procesos físicos que en su conjunto forman los sistemas y como se produce la dinámica de los sistemas mediante el intercambio de mensajes los cuales contienen eventos a simular en un determinado tiempo. A continuación se desarrollarán dos posibilidades para que la simulación avance en el tiempo.

Un aspecto importante a considerar en la simulación es la forma en la cual se generarán en el modelo del sistema los cambios de estado. Para ello se puede considerar el tiempo en el cual ocurre (simulación dirigida por tiempo discreto) o diferenciando estos cambios e identificándolos (simulación dirigida por eventos).

En la simulación dirigida por tiempo discreto, el tiempo de simulación es avanzado en pasos de tiempo de tamaño constante. Para mejorar la precisión se puede reducir el tiempo del paso pero como contrapartida la duración de la simulación aumenta. Una desventaja que presenta es la posibilidad de tener un algoritmo ineficiente cuando los eventos se hallan dispersos temporalmente en forma irregular.

Un algoritmo de simulación dirigido por tiempo consta de los siguientes elementos (estructuras de datos) [47] :

1. Reloj. Por cada avance de tiempo en el reloj la simulación también avanzará.

2. Conjunto de variables de estado. En estas se encuentran los valores que representan los diferentes estados de los procesos físicos. Además de esta información, las variables almacenan en forma implícita o explícita los mensajes recibidos pendientes de ser consumidos que representan sucesos planificados para el futuro.

Junto con el avance del reloj se examinan las variables de estado y se consumen los mensajes con tiempo de ocurrencia iguales al del reloj.

Dependiendo de las características del algoritmo a simular pueden presentarse ciertas situaciones que llevan a una ineficiencia en la simulación. Estas situaciones pueden ser:

- Dispersión en el tiempo de los eventos.

- Cantidad reducida de tarea a realizar entre eventos.

La simulación discreta dirigida por eventos puede ser una alternativa frente a estos puntos débiles de la simulación dirigida por el tiempo.

La simulación discreta dirigida por eventos puede definirse como un modelo de un sistema físico que cambia de estados sólo en instantes discretos de tiempos y que está controlado por eventos. La observación del sistema se realiza únicamente durante la ocurrencia del

evento [24]. A este tipo de simulación se la llama Simulación de eventos discretos (Discrete Event Simulation, DES).

Al igual que en la simulación dirigida por el tiempo existe un reloj (Virtual Time, VT) y un conjunto de variables de estado. A estos elementos se les agrega una nueva estructura de datos, la lista de eventos (Event List, EVL). Esta estructura almacena todos los mensajes planificados para el futuro en forma de mensajes <*evento, tiempo de ocurrencia*> que son los intercambiados por los PFs junto con información de su destinatario. Estos mensajes serán consumidos cuando el reloj alcance el valor de tiempo de ocurrencia del próximo mensaje que se encuentre listo para ser consumido. Esta lista de sucesos se mantiene ordenada por tiempo de ocurrencia en orden creciente.

La secuencia en cada paso de simulación es la siguiente:

- Extracción del mensaje con menor tiempo de ocurrencia.
- Avance del reloj hasta alcanzar el tiempo de ocurrencia del mensaje extraído.
- Simulación de su entrega en el sistema físico donde es consumido.

Esta secuencia puede observarse en la figura 6 [1]. El reloj global está indicando el tiempo 3 y el próximo mensaje con tiempo de consumo más cercano a este tiempo es el 3<e1,5>. Por lo tanto el próximo mensaje a consumir será éste y hará que el reloj global avance hasta el tiempo 5. Como consecuencia se generarán dos nuevos mensajes y la eliminación de uno que existía previamente.

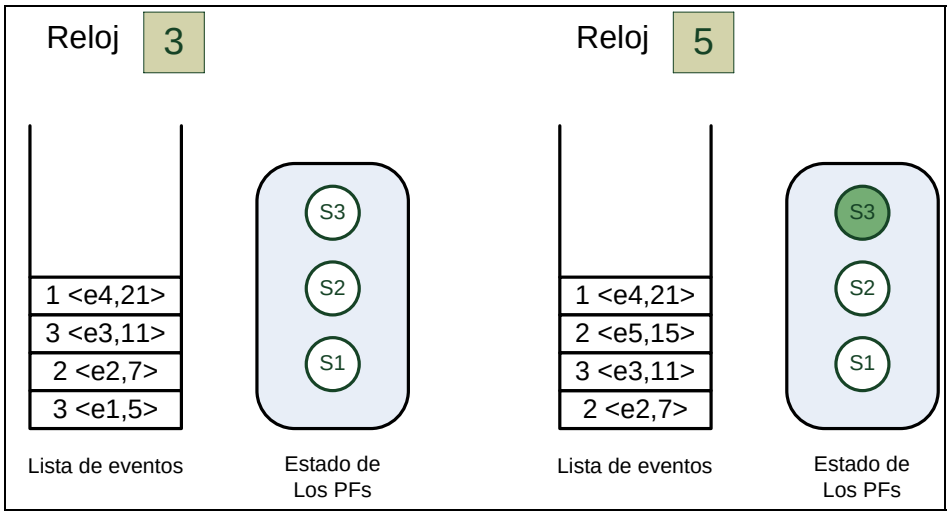


Figura 6: Secuencia de eventos en la simulación

La diferencia entre la simulación dirigida por tiempo y la simulación dirigida por eventos radica en la forma en la que se avanza el reloj. En la simulación dirigida por eventos el reloj avanza tantas unidades de tiempo como sean necesarias para alcanzar el tiempo de ocurrencia del mensaje extraído de la lista de eventos. Esta es una diferencia con la simulación dirigida por tiempo donde el avance del reloj se hace en pasos de una unidad de tiempo.

2.3. Aceleración en la simulación

El constante avance en la complejidad de las simulaciones y las problemáticas que se desean abordar implica un planteamiento diferenciado con el objeto de disminuir los tiempos de procesamiento. Una solución actualmente viable, desde el punto de vista tecnológico, es la paralelización/distribución del simulador.

Simulación paralela y distribuida

En la paralelización de una simulación pueden plantearse dos opciones. Una primera forma de proceder es lanzar tantas simulaciones en paralelo como procesadores se dispongan. Con esta metodología se puede hacer el análisis del comportamiento de un sistema con diferentes conjuntos de valores de los parámetros de entrada. Uno de los problemas que presenta es que cada uno de los elementos de procesamiento debe contar con la suficiente capacidad como para contener el simulador entero. Además puede darse el caso de que alguna o casi todas las simulaciones no sean correctas ya que no todos los valores de entrada tienen por qué ser adecuados [12].

La arquitectura utilizada en el caso mencionado anteriormente es SIMD con el paradigma de programación SPMD. Con este tipo de configuración se consigue una paralelización denominada “paralelismo de datos”. Este paralelismo explota la posibilidad del manejo de grandes cantidades de parámetros de entrada sobre un único código pero no hace uso del potencial paralelismo existente en el código. La distribución de datos sobre las distintas máquinas y la ejecución del mismo código son dos aspectos explotados en este tipo de simulación paralela.

Otra alternativa es utilizar varios procesadores trabajando cooperativamente para acelerar una única simulación (se distribuye una única simulación) y no para hacer varias simulaciones simultáneamente. Esta forma de paralelización se consigue distribuyendo el algoritmo sobre los distintos procesadores. El desarrollo de simuladores de este tipo (simuladores distribuidos)

implica un análisis diferente de los algoritmos. La administración de las listas de eventos, el reloj y la comunicación entre procesos tienen que ser rediseñados generándose un incremento en la complejidad, el diseño y la depuración respecto a los simuladores paralelos.

Simulación paralela vs distribuida

Una importante distinción en las máquinas paralelas o multiprocesadores es su principio de operación. En un entorno SIMD, un conjunto de procesadores ejecutan operaciones idénticas con diferentes datos. Cada procesador procesa su propia memoria local para datos privados y programas, y ejecuta flujos de instrucciones controladas por una unidad central. Por otra parte, los datos pueden variar desde un simple "datum" hasta un conjunto de datos complejos [24].

Las máquinas SIMDs se encuentran implementadas físicamente en arquitecturas con memoria compartida o distribuida. La red de interconexión estática sirve como elemento de intercambio de mensajes (por ejemplo clusters). Cuando el sincronismo impuesto por el principio de operación SIMD es explotado para conducir la simulación con P procesadores (bajo un control central) se habla de simulación paralela.

Un diseño alternativo a SIMD en computación paralela es el modelo MIMD. Un conjunto de procesos son asignados a diversos procesadores que operan asincrónicamente en paralelo. El tipo de comunicación empelado usualmente es el paso de mensajes. En contraste con SIMD, la comunicación en un MIMD tiene el propósito de intercambio de datos. La sincronización local de las actividades de los procesos también se realiza utilizando paso de mensajes. La generalidad de los MIMDs presentan ciertas complejidades adicionales. Cuando un simulador de este tipo es diseñado, implementado y ejecutado, se necesita una codificación explícita de las estrategias de sincronización en el programa paralelo de simulación. Cuando se habla de simulaciones con estrategias de sincronización usando P procesadores y con una codificación explícita se está refiriendo a simulación distribuida.

Niveles de Paralelismo/Distribución

En base a lo mencionado anteriormente se puede hacer una clasificación según el nivel de paralelismo o distribución [24]:

1. Nivel de aplicación: una mayor aceleración en la simulación se puede experimentar con grandes valores de entrada. Esto sucede cuando se asignan réplicas independientes

del mismo modelo de simulación con distintos parámetros de entrada para los procesadores disponibles. La eficiencia en estos casos es elevada y el mismo código de simulación puede ser reutilizado evitando el costo de la distribución (paralelización) del programa pero la escalabilidad es, en teoría, ilimitada.

2. Nivel de subrutina: se distribuyen copias de las subrutinas que constituyen el programa sobre los procesadores de manera de acelerar el evento o el procesamiento de datos.

3. Nivel de componente (sistema físico): ninguna de las dos distribuciones mencionadas anteriormente hacen uso del posible paralelismo disponible en el sistema físico modelado. Con el fin de obtener un mayor beneficio del paralelismo existente, el modelo simulado es descompuesto en modelos de componentes o submodelos. Esta descomposición refleja directamente el paralelismo inherente del modelo o al menos conserva la posibilidad de alguna ganancia durante la simulación.

4. Nivel de eventos:

    a. Lista de eventos centralizados: en este esquema la lista de eventos es una estructura de datos centralizada y administrada por un procesador maestro (master). La aceleración se puede lograr distribuyendo eventos concurrentes a un "pool" de procesadores esclavos (slaves) dedicados a ejecutarlos. Los eventos procesados en paralelo son típicamente los que se encuentran localizados en el mismo momento de tiempo del plano espacio-tiempo.

    b. Lista de eventos descentralizada: los eventos provenientes de puntos arbitrarios del plano espacio-tiempo son asignados a diferentes procesadores, cada uno de una manera regular o estructurada. Un grado mayor de paralelismo puede ser esperado si se explotan las estrategias de simulación que permiten la simulación concurrente de eventos con diferentes tiempos de ocurrencia.

    Esquemas que sigan esta idea requiere protocolos para la sincronización local, la cual tiene a su vez como consecuencia el incremento de la comunicación. El costo de las comunicaciones varía dependiendo de la dispersión de los eventos sobre el espacio y el tiempo en el modelo de simulación subyacente.

Una situación a tener en cuenta en la simulación paralela y distribuida son los efectos de la dependencia de los eventos. El paralelizar una simulación dirigida por eventos es una tarea un tanto difícil ya que por naturaleza este tipo de algoritmos es secuencial. En cada ciclo de simulación se toma el evento planificado para ese momento y se lo simula. Esto puede traer como consecuencia la planificación de eventos para el futuro. La dependencia causa-efecto entre los eventos puede ser fuerte, lo que impide la simulación de varios eventos en paralelo.

A continuación se desarrollará un ejemplo en el que se plantea el problema de la dependencia al paralelizar un algoritmo con un modelo de lista de eventos centralizada [1]:

> Si los primeros mensajes de la lista de eventos son $m_1=<e_1,t_1>$ y $m_2=<e_2,t_2>$, siendo m_i mensajes, e_i eventos y t_i los tiempos de ocurrencia con $t_1<t_2$. El algoritmo secuencial consumirá en primer lugar el mensaje m_1. La simulación de e_1 generará entre otros efectos un nuevo mensaje $m_3=<e_3,t_3>$, con $t_3<t_2$. El siguiente mensaje que será elegido para ser consumido es m_3 y al simular el evento e_3 el suceso e_2 se cancela por lo que se elimina de la lista y nunca será simulado. Ahora bien, si se intenta simular los sucesos e_1 y e_2 en paralelo, luego de hacerlo se simulará el evento e_3 el cual intentará cancelar la simulación del evento e_2 que ya ha sido simulado. En consecuencia, se presenta un problema de causalidad. Esta es una situación no deseada que no se puede permitir que suceda.

2.4. Causalidad

En la simulación dirigida por eventos, es indispensable que se respete la dependencia temporal para que la simulación de un evento no pueda afectar otros simulados anteriormente.

Los distintos tipos de dependencias entre eventos pueden definirse de la siguiente forma [1]:

- Un evento e_i afecta al evento e_j si al menos alguna de estas condiciones se verifica considerado que $t_{ei} < t_{ej}$:
  - La simulación de e_i hace que se genera la planificación de e_j.
  - La simulación de e_i hace que se cancele e_j, planificado con anterioridad.
  - La simulación de e_j exige leer o escribir información de estado que fue creada o alterada durante la simulación.

- Un evento *b* depende causalmente de un evento *a* si existe una secuencia de sucesos $a = e_0, e_1, \ldots\ldots, e_n = b$ tal que para cada par (e_i , e_{i+1}) ocurre que e_i afecta a e_{i+1}.

- Para cualquier par de eventos, *a* y *b*, puede darse el caso que ni *a* dependa causalmente de *b* ni al revés; en consecuencia estos sucesos son causalmente independientes.

En consecuencia, eventos con el mismo tiempo de ocurrencia son siempre causalmente independientes. De aquí surgen dos posibles paralelismos: el primero estará determinado por la simulación simultánea de eventos con el mismo tiempo de ocurrencia evitándose de esta manera los problemas de causalidad. Teniendo como inconveniente es que el paralelismo extraído en general es escaso.

El segundo tipo de paralelismo consistirá en la simulación simultánea de eventos que no tienen el mismo tiempo de ocurrencia. La limitación en el orden es solo causal y no temporal y de esta forma se permite extraer mayor paralelismo. Por lo tanto para garantizar que una simulación es correcta basta con mantener un orden causal en la simulación de los eventos, coincida o no con la temporal.

En la figura 7 se muestra un ejemplo de como podría paralelizarse una simulación secuencial en base a las restricciones causales mencionadas anteriormente [1].

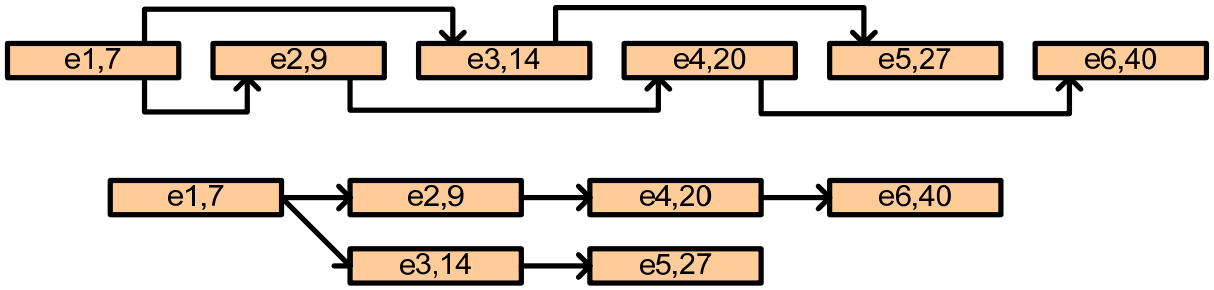


Figura 7: Ejemplo de paralelización

2.5. Paralelización de la simulación dirigida por eventos

Las estrategias de simulación con listas de eventos distribuidas tienen como objetivo la división del sistema global en un conjunto de procesos lógicos (Logical Processes, LP), uno por cada PF. Con esto, se procura explotar al máximo el paralelismo inherente entre los respectivos componentes modelados con la ejecución concurrente de estos procesos. De esta forma, la simulación puede verse como la cooperación de un arreglo de LPs interactuando, cada uno simulando un subespacio del espacio-tiempo el cual se llamará región. En general,

una región es representada por el conjunto de todos los eventos en un subtiempo del tiempo de simulación, o el conjunto de todos los eventos en un cierto subespacio del espacio de simulación [24] [63].

En la figura 8 se observa la arquitectura de la simulación de procesos lógicos donde se puede considerar [24]:

- Un conjunto de LPs es diseñado para ejecutar la ocurrencia de eventos sincrónicamente o asincrónicamente en paralelo.

- Un sistema de comunicaciones (CS) provee a los LPs los medios para intercambiar datos locales pero también para sincronizar las actividades locales.

- Cada LP_i tiene asignada una región R_i, como parte del modelo de simulación. Sobre esta región el "motor de simulación" (SE_i) trabaja en modo dirigido por eventos ejecutando eventos locales (y genera remotos). De este modo progresa el reloj local (Local Virtual Time, LVT).

- Cada LP_i (SE_i) tiene acceso solamente a una partición estática del subconjunto de variables de estado $S_i \subset S$, disjunta a las variables de estado asignada a otros LPs.

- Dos tipos de eventos son procesados en cada LP_i:

    - Los eventos internos que tienen impacto causal solamente para $S_i \subset S$.

    - Los eventos externos que también afectan $S_j \subset S$ ($i \neq j$), los estados locales de otros LPs.

- Una interfaz de comunicación (CI_i) en el SE "protege" de la propagación de los efectos causales de los eventos a ser simulados por LPs remotos.

La forma en la que la CI avanza en la simulación de los eventos y el tiempo, determina el tipo de simulación que pude ser:

- Conservativa.

- Optimista.

La CI es la responsable de prevenir o controlar el incumplimiento de la causalidad. En el caso de las conservativas, la CI inicia al SE solamente cuando está seguro que no habrá errores de causalidad. En el caso de los optimistas, la CI inicia al SE y en caso de encontrar incumplimientos de causalidad se retoma la simulación al punto en el cual la causalidad se respetaba.

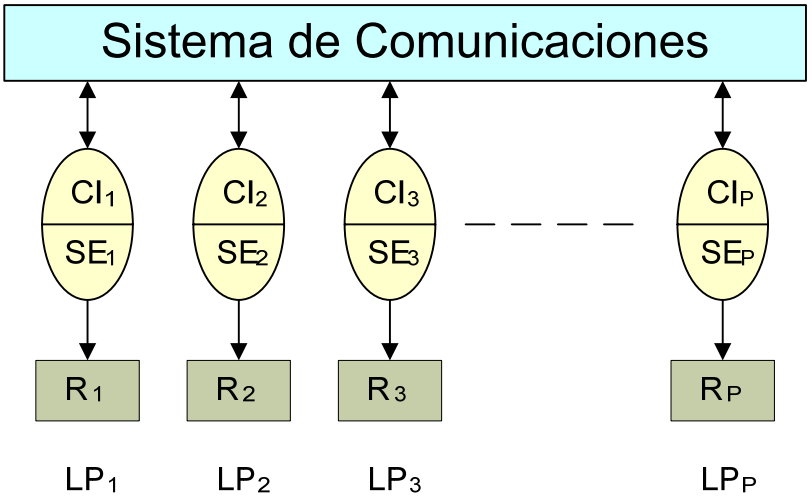


Figura 8: Arquitectura de la simulación de procesos lógicos

Método de simulación conservativo.

Los trabajos de Chandy, Misra [20] y Bryant [15] fueron los pioneros en el tema de estrategias conservadoras. A menudo se la suele llamar protocolos CMB (Chandy-Misra-Bryant).

Esta estrategia respeta las relaciones de causalidad entre los eventos forzando a los PLs a no consumir un mensaje hasta que no exista la certeza de que es imposible que llegue otro mensaje con tiempo de ocurrencia (timestamp) menor. Esto se consigue, en parte, mediante el envío a través de los LPs, de mensajes (externos) del tipo <ee@t>, donde ee es el evento y t es una copia del LVT del LP que lo envía en el instante en el cual el mensaje fue creado y enviado y donde t=ts(ee) es también llamado el tiempo de ocurrencia del evento.

Un LP que sigue el método conservador permite solamente procesar eventos “seguros”. Además, todos los eventos, internos y externos, deben ser procesados en orden cronológico. Esto garantiza que la cadena de mensajes producidos por un LP es “entregado” en orden cronológico, y un sistema de comunicación (figura 9) preserva el orden de los mensajes enviados desde LP_i a LP_j (FIFO) es suficiente para garantizar que no lleguen fuera de orden

cronológico mensajes en algún LP_i . Una simulación conservativa puede ser entonces vista como el conjunto de todos los LPs $LP = \bigcup_k LP_k$ conjuntamente con un conjunto de canales de comunicación FIFO $CH = \bigcup_{k,i(k\neq i)} ch_{k,i} = (LP_k, LP_i)$ que constituye el grafo de procesos lógicos (Graph of Logical Processes, GLP) GLP = (LP,CH). Es importante notar que GLP tiene una topología estándar, la cual no permite la planificación dinámica de LPs en un conjunto de procesadores físicos.

La interfaz de comunicación CI de un LP mantiene un buffer de entrada IB[i] y un reloj del canal (o link) CC[i] para cada canal $ch_{i,k}$ perteneciente a CH con referencia a LP_k, (figura 9) [24].

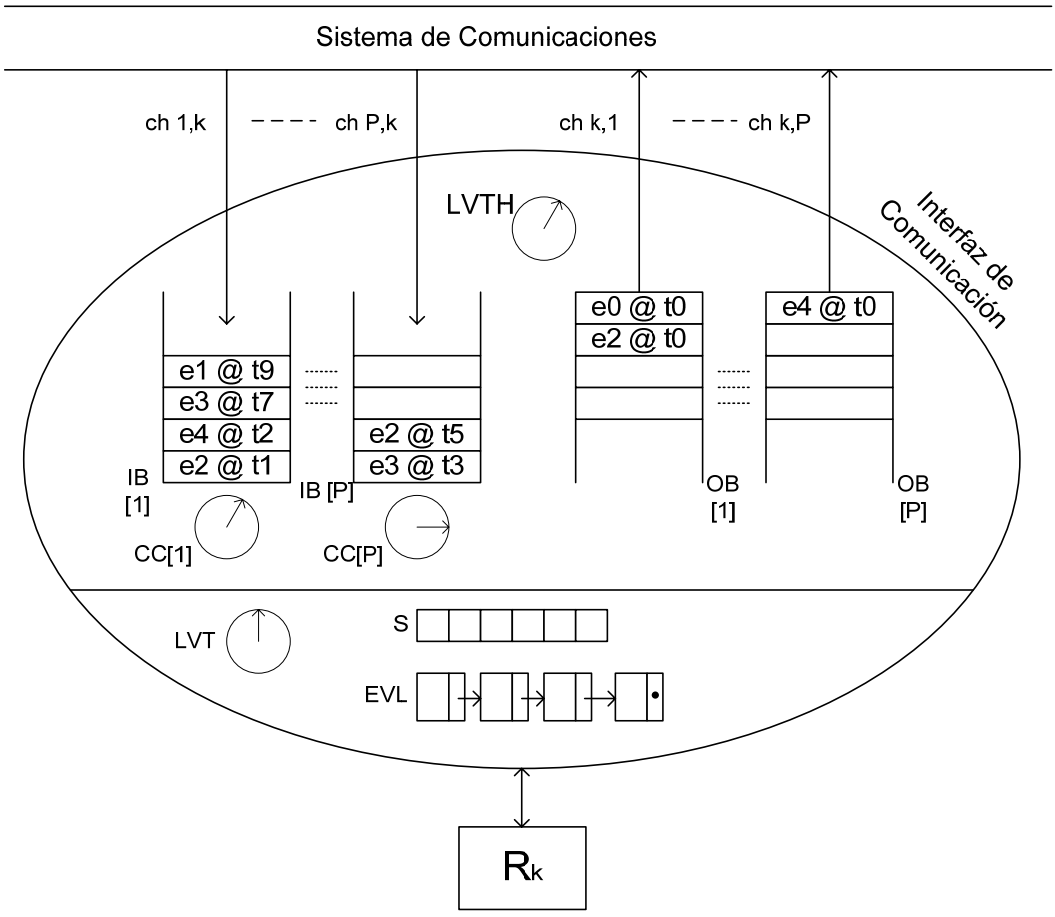


Figura 9: Arquitectura de un proceso lógico conservador

IB[i] almacena los mensajes recibidos en orden FIFO, mientras CC[i] mantiene una copia del tiempo de ocurrencia del mensaje de la cabeza del IB[i] y CC[i] inicialmente es puesta a cero. LVTH = $\min_i$CC[i] es el horizonte de tiempo mayor hasta el cual LVT es permitido progresar simulando eventos internos o externos, puesto que eventos externos con tiempo de ocurrencia (timestamp) menores que LVTH no podrán ser recibidos.

CI inicia el SE como un SE dirigido por evento basado en el proceso de eventos (internos) en el EVL, pero también para procesar eventos (externos) de las correspondientes IBs respetando el orden cronológico y solo hasta que el LVT encuentra el LVTH. Durante esto, SE puede haber producido futuros eventos para LPs remotos.

Por cada uno de los eventos futuros se crea un mensaje agregando una copia del LVT para el evento, y depositándolo en el FIFO de salida (output buffer OB[i]) para que sea distribuido por el sistema de comunicación. CI mantiene OB[i] individuales para cada canal de salida $ch_{k,i}$ perteneciente CH para subsecuentes LPs LP_i.

Si dentro del horizonte LVTH no hay ningún evento interno o externo para ser procesado entonces los LP_k bloquean su procesamiento, y no reciben nuevos mensajes que ensanchan el horizonte de tiempo.

La política de "blocking-until-safe-to-process" puede generar dos problemas, el interbloqueo (deadlock) y el agotamiento de memoria (memory overflow). El primero puede generarse por ejemplo cuando un PL se encuentra a la espera de mensajes de otro LP que ya ha terminado su simulación.

Otra fuente básica de interbloqueo es la formación de ciclos de LPs. En esta situación cada LP perteneciente al ciclo está bloqueado esperando mensajes de otros. Un nuevo problema que se puede presentar en un interbloqueo es que los LPs que no pertenecen al ciclo les envíen mensajes a los LPs bloqueados y hagan crecer en forma impredecible sus IB y generen agotamiento de memoria (es posible incluso en ausencia de interbloqueo) [2].

En la figura 10 se puede ver un ejemplo clásico [24] de interbloqueo en una simulación en el cual ninguno de los tres procesos que intervienen puede avanzar sus respectivas simulaciones, ya que todos esperan por sus vecinos.

En el caso general los deadlock suelen aparecer cuando un ciclo de IQ vacías se produce con sus relojes asociados situados en tiempos pequeños. Normalmente los deadlocks suelen generarse frecuentemente cuando hay pocos eventos no procesados comparados con el número de links en la red o si los eventos no procesados se encuentran agrupados en una parte de la red.

Con respecto al desbordamiento de memoria, este puede ocurrir cuando la simulación no avanza constantemente y existen procesos demasiado lentos que no les dé tiempo a procesar

los eventos tan rápido como se generan o llegan. Esta situación puede provocar un crecimiento descontrolado de los buffers de entrada y de la lista de eventos, causando errores de desbordamiento de la memoria. Además estos problemas de memoria se complican cuando existen algunos procesos lógicos en interbloqueo (y por lo tanto no pueden procesar eventos), mientras el resto de los LPs les continúan enviando eventos.

Una forma de evitar el problema de interbloqueo se puede encontrar en [47] en la que se envían mensajes nulos (no tienen acción sobre el sistema simulado) para sincronizar y ampliar el LVTH. En [17] [27] [47] [70] se pueden encontrar diversas técnicas como la "detección y recuperación" (detection&recovery) o evitar (avoidance) entre otras, para solucionar el problema de interbloqueo.

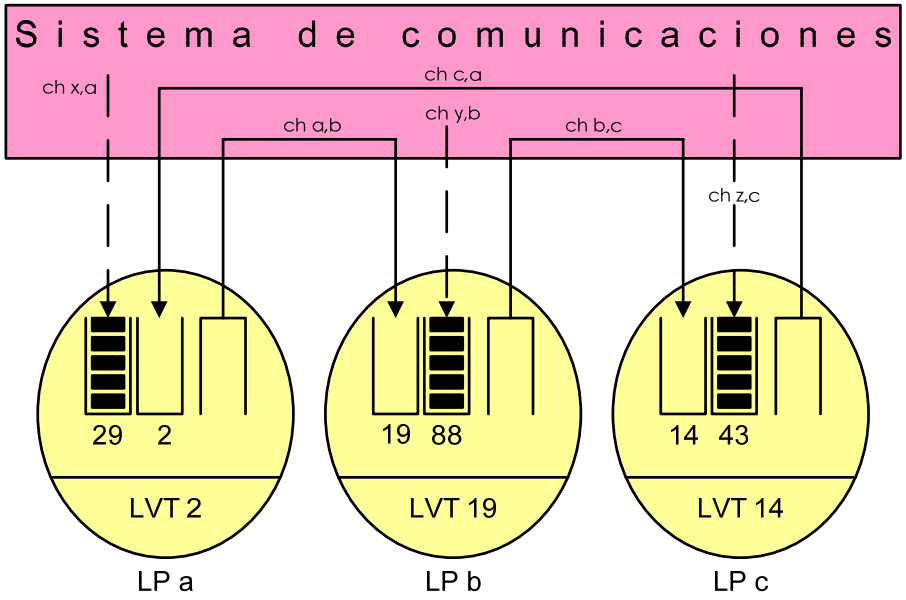


Figura 10: Interbloqueo y agotamiento de memoria

Método de simulación optimista

El mecanismo Time Warp [38] es el más conocido y utilizado de los protocolos optimistas. Este algoritmo detecta los errores de causalidad cada vez que recibe un mensaje con un tiempo de ocurrencia menor que su reloj virtual. Esto es, que el evento externo está realizando una referencia el pasado del proceso lógico (no se respeta el orden causal).

La arquitectura básica de un proceso lógico optimista que utilice el mecanismo de rollback es similar al algoritmo del protocolo conservador CMB pero con un único canal de comunicación.

Como se puede observar en la figura 11 los procesos interactúan con los demás LPs de la simulación a través de un sistema de comunicaciones. La función de este sistema es asegurar la recepción y emisión de los mensajes entre los procesos lógicos.

Si bien el funcionamiento básico de este sistema es idéntico en ambos protocolos, estos difieren en los requisitos que le exigen. Los métodos conservativos requieren que el sistema de comunicaciones les asegure que los mensajes llegarán en el mismo orden en que se enviaron. Este requisito no es necesario en el protocolo Time Warp ya que no necesita asegurar una secuencia estricta en el procesamiento de los eventos y por lo tanto no necesita que estos lleguen en orden.

Al igual que en el protocolo conservativo el Time Warp utiliza un buffer de entrada (IB) y un buffer de salida (OB) como interfaz entre el motor de simulación y el sistema físico de comunicaciones.

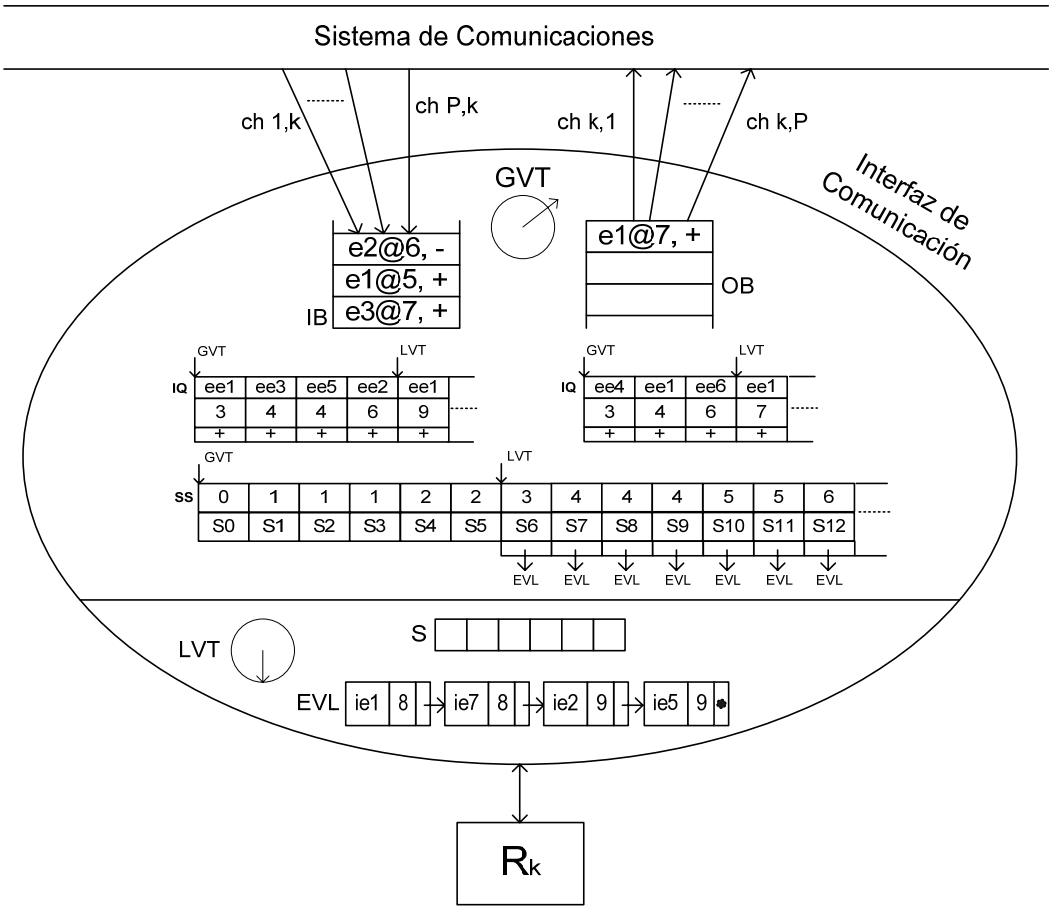


Figura 11: Arquitectura de un proceso lógico optimista

En el mecanismo Time Warp la sincronización con el resto de los LPs ya no se realiza en los buffers de entrada, si no que se ha desplazado al motor de simulación. A diferencia del método conservador donde es necesario sincronizar los distintos canales de entrada, aquí no es necesario separarlos para tal fin. Por lo tanto el Time Warp sólo utiliza un único buffer de entrada para todos los mensajes externos (independientemente de donde provengan) y un buffer de salida para los mensajes enviados a otros LPs. La utilización de un único buffer de entrada o salida significa que la interfaz de comunicaciones no tiene información disponible sobre origen o destino del mensaje. Como esta información es imprescindible para el funcionamiento del protocolo deben estar integradas en el propio mensaje.

El motor de simulación consiste en una simulación dirigida por eventos discretos, que consta de una lista de eventos que hay que procesar (EVL), un reloj local (LVT) que indica en que momento de la simulación se encuentra el proceso lógico y por último el estado de simulación.

Las mayores diferencias entre los métodos conservativo y optimista radican en al forma que este último implementa la detección y recuperación de los errores de causalidad.

Dentro del sistema de detección y recuperación de errores se pueden encontrar tres subsistemas diferentes que interactúan entre ellos: los sistemas de detección, recuperación y propagación de errores de causalidad.

- **Subsistema de detección de errores de causalidad**. Es el encargado de soportar la detección de los errores de causalidad que se puedan producir. Para poder detectar estos errores es imprescindible llevar un histórico de todos los mensajes externos recibidos. Este histórico se denomina cola de entrada (Input queque, IQ).

  En la cola de entrada se guardan todos los mensajes recibidos por el LP. Estos mensajes pueden haber sido ya procesados o no. Cada vez que se procesa un evento del buffer de entrada, este se almacena cronológicamente en la cola de entrada.

- **Subsistema de recuperación de errores de causalidad**. Este subsistema debe permitir volver a cualquiera de los estados por los cuales ha pasado la simulación. Para poder volver hacia atrás la simulación (rollback) es necesario recuperar por un lado el estado del sistema físico simulado en ese momento (estado local de la simulación) y por otro lado recuperar el estado del Time Warp. Por lo tanto, para

poder realizar esta operación, es necesario salvar periódicamente el estado de simulación del proceso lógico por si es necesario realizar un rolback.

Los estados de simulación salvados se gestionan mediante la pila de estados (State Stack, SS). Cada una de las entradas de la pila de estados contiene el instante de tiempo al cual pertenece (LVT) y una copia completa del estado del motor de simulación.

Por otro lado para conseguir recuperar satisfactoriamente un determinado instante del estado de simulación, también es necesario recuperar los estados que tenía en ese momento la cola de entrada y la cola de salida. Esta operación se puede realizar sobre las mismas colas, mediante la eliminación de todos aquellos mensajes pertenecientes al mismo proceso que ha provocado el rolback y que tenga una marca de tiempo mayor del tiempo del estado que se tiene que recuperar.

El tamaño de la pila de estados podría llegar a ser considerable si se tuvieran que guardar todos los estados por los cuales ha ido pasando la simulación. Para evitar esto se utiliza el tiempo virtual global (Global Virtual Time, GVT). El GVT es una marca de tiempo de simulación que identifica a partir de que instante de tiempo la simulación está en un estado seguro. Si el GVT está bien calculado no se pueden producir errores de causalidad con un tiempo menor a él. El cálculo del GVT se obtiene mediante el menor de todos los tiempos locales virtuales de todos los LPs.

El Time Warp utiliza el GVT para saber el intervalo del tiempo de simulación (GVT,LVT] en el cual se puede producir errores de causalidad. Por lo tanto el subsistema de recuperación de errores sólo necesita mantener almacenados en la SS aquellos estados de simulación situados dentro de dicho intervalo.

- **Subsistema de propagación de errores de causalidad**. Por último el sistema de propagación de errores es el encargado de anular todos los eventos externos que ya han sido enviados y que han sido invalidados por un error de causalidad.

  Para poder realizar la eliminación de los eventos externos se necesita llevar un control de todos los mensajes que han enviado el proceso lógico. Este histórico se realiza en la cola de salida, la cual tiene que contener el evento y el LVT de todos los mensajes enviados.

A la hora de invalidar un mensaje externo, el TW envía un duplicado del mismo mensaje pero con el signo negativo. Este tipo de mensaje específico de los protocolos optimistas se denomina antimensaje. El mecanismo de anulación de eventos externos utiliza los antimensajes para propagar los errores de causalidad y su recuperación para todos los procesos lógicos [18].

El método optimista permite explotar todo el paralelismo inherente a la aplicación. La utilización de ciertos modelos puede provocar la disminución de los tiempos de simulación como consecuencia de la generación de cadenas de rollbacks necesarias para garantizar la causalidad en el sistema.

La cantidad y la administración de la memoria requerida para almacenar los estados pasados de la simulación es otro de los factores que pueden considerarse negativos en la simulación optimista. Soluciones y discusiones sobre ambos problemas pueden encontrarse en [9] [41] [57] [61] [44] [29] [28] [66] [5].

Capítulo 3

Modelado de Sistemas Biológicos

3.1. Introducción

El modelado de la dinámica de poblaciones o ecosistemas es de gran interés para biólogos y ecólogos. Estos modelos les son útiles, por ejemplo, para analizar el comportamiento de especies o poder predecir el movimiento de las poblaciones en los ecosistemas.

La complejidad de los modelos biológicos se encuentra estrechamente ligada al nivel de abstracción del sistema generado en el momento del modelado. En general, un aspecto muy importante a considerar es la cantidad de individuos involucrados en el sistema o la cantidad de especies que forman parte del ecosistema. El número de individuos influye en el tiempo requerido para que el simulador pueda entregar los datos esperados de la simulación. La cantidad de especies que participan en el ecosistema incrementan la complejidad del desarrollo del simulador puesto que se tiene que modelar e implementar no solo la interacción entre individuos de la misma especie sino también entre individuos de distintas especies.

La resolución de un problema biológico o el estudio de un ecosistema o parte de él puede ser realizado mediante la utilización de la simulación ya sea como herramienta para predecir el comportamiento o como uno de los tres elementos que forman la ciencia computacional.

Los otros dos elementos que la integran son las distintas disciplinas de la ciencia como ser la química, la biología y las ciencias aplicadas como lo es la matemática.

En este capítulo se realizará una introducción de los principales modelos utilizados en biología, los orientados a la población o modelos de variables de estado y los modelos orientados al individuo (en la literatura pueden encontrarse también como modelos basados en el individuo).

El modelo utilizado en la presente tesis, Fish School, es un modelo orientado al individuo que describe el movimiento de peces tomando como elemento fundamental el individuo. El comportamiento grupal es consecuencia de la interacción entre los diversos integrantes del banco de peces. Para este modelo se he desarrollado prestando atención tanto al modelo biológico como el matemático. En las secciones siguientes se describen las modificaciones y generalizaciones hechas para adaptar el modelo bidimensional definido en la literatura a un modelo en el espacio.

3.2. Modelos biológicos

La mayoría de los modelos matemáticos en biología, que van desde simples ecuaciones de crecimiento poblacional hasta funciones de descripciones complejas de ecosistemas, están basados en suposiciones que incumplen dos principios básicos de la biología [35]:

- Cada individuo es diferente.
- Las interacciones son inherentemente locales.

A menudo los modelos combinan muchos organismos individuales y asumen que ellos pueden ser descritos por una única variable, como por ejemplo, el tamaño de la población. Este procedimiento rompe con el principio biológico de que cada individuo es diferente, con un comportamiento y fisiología que resulta de una única combinación de genética e influencias medioambientales.

Muchos modelos no hacen distinción entre los distintos sitios donde podrían encontrarse los individuos. Se asume que cada individuo produce un efecto igual en cada uno de los otros individuos. Asumir esto no cumple con el principio biológico de que las interacciones son inherentemente locales. Un organismo es afectado, ante todo, por otros organismos con los cuales entra en contacto.

Estas problemáticas han sido reconocidas durante muchos años por ecologistas pero las formas efectivas de desarrollar modelos que las evitaran eran pocas. Los primeros modelos desarrollados para estudiar los sistemas ecológicos, llamados modelos de variables de estado (state-variable models) o modelos orientados a la población, contaban con los dos problemas mencionados anteriormente.

Teniendo en cuenta estas limitaciones, los primeros trabajos realizados para describir la dinámica poblacional se efectuaron a través de ecuaciones que describen el comportamiento de los individuos mediante distribuciones continuas de las características individuales en las poblaciones. La utilización de estas ecuaciones a través de un análisis directo o bien con modelos computacionales simples pueden ser aplicados para representar estos comportamientos. Entre tanto, los modelos basados en esta solución sólo pueden representar en forma adecuada a un grupo reducido de cientos o miles de individuos.

Con el crecimiento rápido del poder computacional de los últimos años, se han desarrollado programas capaces de reproducir el comportamiento de los miembros de poblaciones de forma individual. Sin embargo, los modelos de variables de estado se han mantenido como la principal herramienta para el modelado en la teoría ecológica.

El individuo es una unidad lógica básica para el modelado de fenómenos ecológicos. Aunque todos los modelos requieren tomar algún tipo de simplificación, los modelos basados en el individuo no requieren de las mismas abstracciones asumidas en el uso de modelos de variables de estado.

Uno de los aspectos más significativos de los modelos orientados al individuo es que permiten investigar diversos tipos de cuestiones que han sido difíciles o imposibles utilizando las aproximaciones de los modelos clásicos.

Los modelos basados en el comportamiento del individuo generan un grado de integración a diferentes niveles en la jerarquía tradicional de procesos ecológicos. Cada nivel (de esa organización) ha sido tradicionalmente tratado como un campo independiente, por ejemplo: ecología del ecosistema, de la comunidad, de la población del comportamiento, de parámetros fisiológicos. Cada campo tiene su propio conjunto de fenómenos para describirlo, y la mayoría cuenta con sus propios modelos, distintos entre sí. Los modelos orientados al individuo demuestran que todos los niveles de la jerarquía pueden ser entendidos en el

contexto de la interacción de los organismos individuales, directa o indirectamente, a través de los efectos resultantes de la interacción en el medioambiente.

La principal ventaja de los modelos orientados al individuo sobre muchos modelos de variables de estado es que las propiedades de los individuos y el mecanismo por el cual ellos interactúan con su medioambiente pueden ser medidos. Estos son los datos que comúnmente son utilizados por científicos de diversas áreas. Los datos pueden ser extraídos para generar los parámetros necesarios para este modelo en contraposición con los modelos de variables de estado para los que los procesos deben ser medidos a gran escala, extrapolando datos de pequeña escala, o derivados de modelos apropiados para obtener los parámetros de entrada adecuados.

Existen diversos trabajos que demuestran que los ecosistemas son grandes sistemas realimentados, dinámicos y que poseen mecanismos propios de autorregulación similares a otros sistemas físicos. Si bien el estudio realizado sobre estos sistemas es importante, no se han conseguido los mismos éxitos en su modelado y simulación que en otras disciplinas como control automático o las redes de tráfico, entre otros. Por ello, los modelos orientados al individuo se han propuesto como alternativas al enfoque clásico de los modelos orientados a población.

El inicio del desarrollo de los modelos de sistemas ecológicos es de principios del siglo XX. Lotka y Volterra [65] han modelado mediante ecuaciones las interacciones entre una presa y un depredador. Estos modelos, que aún son válidos, se basan en leyes de acción de masas y describen el comportamiento de sistemas por medio de ecuaciones diferenciales. Los modelos mencionados dan soluciones óptimas para sistemas de pequeño tamaño pero no pueden ser generalizados para cualquier sistema [39].

Estos modelos presentan dos objeciones:

1. La población es tratada como cantidades de individuos uniformes, con comportamiento común. Además, el modelo suele asumir que la población a estudiar es de tamaño considerable y presenta una distribución de probabilidad normal en lo que concierne a su comportamiento. Por tanto, los individuos podrán ser representados por su valor medio.

2. Por otro lado Lotka y Volterra asumen que los modelos son deterministas, es decir, que para tener una solución es necesario resolver unas ecuaciones de cierta complejidad con unas condiciones iniciales (resolución analítica y/o numérica).

Una solución a la primera objeción son los modelos estructurados por edades (*Age-structured models)* [32], que intentan definir comportamientos diferentes para individuos de distintos grupos de edad. El modelo es más representativo pero continúa asumiendo uniformidad en el comportamiento de los individuos dentro de su grupo de edad.

Respecto a la segunda objeción, una alternativa consiste en desarrollar modelos estocásticos [43], que permiten agregar coeficientes de incerteza en el comportamiento de la evolución a lo largo del tiempo. Estos modelos también pueden resolverse analíticamente (son modelos complejos) pero la solución final está regida por una distribución de probabilidad.

Uno de los principales argumentos en contra de los modelos clásicos, con resolución analítica, es el caos [39] [7]. La teoría del caos ha irrumpido con fuerza en los últimos años y ha contribuido a explicar comportamientos de los cuales no se tenía una justificación hasta el momento.

No se realizará un desarrollo en profundidad sobre que es el caos, pero sí algunas afirmaciones cualitativas. Se dice que un sistema presenta comportamiento caótico cuando (bien que el comportamiento pueda ser obtenido por ecuaciones deterministas, de aquí que se habla de caos determinista) exista una dependencia notoria de las condiciones iniciales y cuando se producen pequeñas variaciones durante los tiempos en que el sistema se ejecuta. Es decir, resolver dos sistemas prácticamente iguales puede dar resultados muy diferentes.

Por extensión, un argumento para justificar la poca utilidad de los modelos clásicos radica en que se ignora cualquier incertidumbre que puede ocurrir durante el tiempo de "simulación", Se sostiene que el sistema depende solamente de condiciones iniciales. Hasta ahora se creía que menospreciar esta incertidumbre conducía a un cierto error final (acotado) que podía ser estudiado y hallado por métodos estadísticos [43]. En un sistema caótico, un error provocado por una mínima perturbación del sistema acostumbra a tener una relación difusa con la causa que la ha provocado, y, por tanto, su análisis se hace prácticamente imposible.

Los sistemas predispuestos a ser caóticos son los sistemas dinámicos realimentados, del tipo de los que se estudian en la construcción de modelos ecológicos. Si bien todos los autores

comparten que el caos está presente en la naturaleza y en los sistemas ecológicos, algunos van más allá y afirman que un sistema ecológico es inherentemente caótico [39] y, por tanto, necesita de un cambio de filosofía en la forma de tratarlo.

3.3. Tipos de modelos biológicos

Modelo orientado al Individuo (IoM).

El concepto de los IoM es describir la conducta de cada individuo por separado (utilizando una serie de reglas), y obtener el comportamiento grupal como la interacción de los individuos. Dos ventajas importantes de este tipo de modelos son:

- El incrementar en un número más grande de especies no implica un salto intelectual, al contrario que en los modelos clásicos donde el crecimiento del modelo lo transformaba en intratable. Se puede afirmar que los IoM son modelos que escalan bien y, por tanto, surge la atractiva idea de construir un modelo de modelos[23]. Es decir, partir de una idea incremental en la construcción de modelos (bottom-up) en lugar de intentar entender el sistema complejo como en uno único (top-down).

- Otra idea que se utiliza para definir este tipo de sistemas [58], es la aparente paradoja que supone la emergencia de comportamientos complejos a partir de organizaciones básicas realmente simples. Esta es la conocida idea de “el todo supera la suma de las partes”, por ejemplo, comportamiento de colonias de termitas o redes neuronales. Cuando lo que se pretende es la simulación de sistemas con una complejidad elevada es más sencillo partir de la base (individuo simple) que describir analíticamente la colectividad (enfoque clásico).

Existen fuertes controversias a la hora de mostrar la utilidad de los modelos orientados al individuo. Algunos ecologistas sostienen que es solamente una forma diferente de atacar el problema (más expresiva, si se quiere) que brinda resultados similares. En [45] se discuten los resultados de simular modelos bajo las dos aproximaciones, y se extraen las conclusiones. De todas maneras, existen dos factores externos, y algunos objetivos que han facilitado la proliferación de los IoM:

- El desarrollo que ha experimentado la informática en los últimos tiempos ha dotado a los IoM de más potencia de cálculo para dar respuesta en tiempos razonables. Si bien

los modelos clásicos también se resuelven numéricamente, (solución numérica de sistema de ecuaciones diferenciales) su complejidad en tiempo será siempre menor al considerar a los individuos como un todo. La complejidad en modelos clásicos es *O(Número de especies)* mientras que en modelos IoM es *O(Número de individuos)*.

- La aparición de los lenguajes orientados a objetos da una expresividad muy elevada a la hora de codificar los simuladores (identificación de los individuos como objetos de unas determinadas clases, herencia, anexar modelos simples para formar uno más complejo…).

En la literatura se pueden encontrar diversos modelos orientados al individuo e implementaciones de simuladores para estos como puede observarse en [40] [22] [42].

La definición de los IoM se basa en reconocer a cada individuo como un agente autónomo que actúa según un conjunto de reglas bioinspiradas. El problema radica, evidentemente, en definir un conjunto de reglas realistas para organismos complejos.

Uno de los problemas de estos tipos de modelos desde hace tiempo, es la falta de un sistema estándar de representación. La mayoría de los grupos de investigación dan descripciones textuales del modelo (inherentemente ambiguas [39]) que dificultan la reproducción a otros grupos. Dada la relación evidente entre el modelado orientado al individuo y las técnicas de programación orientadas a objetos, una buena idea podría ser utilizar las técnicas típicas de análisis y diseño de software orientado a objetos [26] [14] para la especificación de modelos.

Fish Schools sigue los modelos IoM. Basándose en la interacción de los individuos, determina el movimiento del banco de peces. A partir de la observación de comportamiento global se pueden observar en su movimiento figuras complejas que se rigen por reglas básicas del comportamiento de los peces.

Modelo orientado a la población, ecuaciones de Lotka-Volterra

En este tipo de sistemas se modela la población como un todo uniforme, y se hacen las analogías con los modelos físicos (equilibrio de líquidos, movimiento de masa,..) [32] [65].

Un ejemplo sencillo puede observarse en la ecuación (1) que describe el crecimiento de una población "sin depredadores", donde *x* es la densidad de población.

$$\frac{dx}{dt} = rx \tag{1}$$

Esta ecuación diferencial indica cómo la población en un tiempo *t* es directamente proporcional a la que había hasta el momento, modificada por una constante *r* (dependiente de la especie, por ejemplo). El crecimiento de la población es ilimitado ya que no tiene en cuenta que en el entorno los recursos son limitados. La solución para la ecuación es:

$$x = x_0 e^{rt} \tag{2}$$

y, es evidente que

$$\lim_{t \to \infty} x_0 e^{rt} = \infty \tag{3}$$

Por lo tanto, sólo para intervalos limitados de *t* las soluciones son realistas.

Una aproximación mayor a la realidad se puede obtener utilizando las ecuaciones que derivan de los trabajos de Lotka-Volterra [65] [32] que datan de la década de 1920.

Dichas ecuaciones reflejan la expansión temporal de una especie pero, a diferencia de la primera ecuación mostrada anteriormente, en éstas se incluye un nuevo elemento que hace al modelo más próximo a la realidad, los depredadores. Ciertos factores como la edad, la disponibilidad de alimentos para las presas y las cuestiones energéticas no son tenidas en cuenta.

Las siguientes ecuaciones diferenciales resumen dicho modelo:

$$\begin{cases} \frac{dx}{dt} = ax - bx^2 - cxy \\ \frac{dy}{dt} = -ey + c'xy \end{cases} \tag{4}$$

donde *x* e *y* son las densidades de presas y depredadores respectivamente. *a,b,c,c',e* son constantes del sistema modelado con $a>>b$.

De la primera ecuación, se extrae que la población de presas crecerá proporcionalmente a *ax*. El término $-bx^2$ limita el crecimiento desmesurado, estabilizando el crecimiento. Por último, el término *-cxy* representa la interacción con los depredadores, es decir, que el crecimiento de la población de presas es inversamente proporcional a la de los depredadores. De no existir éstos últimos, solamente conseguiría un efecto de crecimiento que para valores altos de población se estabilizaría por el término $-bx^2$.

En la segunda ecuación se observa que el número de depredadores disminuye en *-ey* cuando no hay presas debido a la ausencia de alimento. El crecimiento de la población es directamente proporcional al número de presas (en *c'xy*) puesto que la existencia de alimento permite el aumento del número de depredadores.

El sistema descrito se comporta como un típico sistema realimentado: estable, casi siempre oscilante o inestable, con la extinción de presa o depredador en alguno de los flancos de la oscilación.

Los sistemas orientados a población acostumbran a plantearse como ecuaciones continuas, si bien también pueden representarse discretamente [32] y la equivalencia es biunívoca. En la naturaleza, el sistema es discreto si se modela la cantidad de individuos, (la población siempre varía en unidades de individuos), pero para cantidades grandes, se puede considerar que la población se mueve en un rango continuo, y, por tanto, se puede representar por ecuaciones diferenciales.

3.4. Fish Schools

Diversos grupos de animales presentan un comportamiento jerárquico. En el caso de los bancos de peces, en general, no se encuentra un comportamiento de este tipo. El tamaño, forma y estructura de un banco de peces depende de la especie e inclusive, pueden variar para una misma especie a lo largo del tiempo. Si bien el comportamiento general de los individuos del banco es similar, pueden existir variaciones entre ellos debidas a la edad de los miembros. A pesar del factor de la edad, se pueden encontrar en la literatura diversos patrones típicos de comportamiento.

Durante los períodos de alimentación y descanso, la orientación de los peces muestra un patrón marcadamente aleatorio. En contraposición a este comportamiento, durante el desplazamiento, el patrón de movimiento es altamente paralelo. Cuando un banco de peces es atacado por un depredador, el alineamiento paralelo cambia, tomando diversas formas como la división del grupo. Esto demuestra la existencia de una buena coordinación en el comportamiento del grupo sin el dominio de algunos individuos sobre otros.

En la literatura existen diversos trabajos relacionados con el modelado del movimiento de peces [4]. Algunos de estos modelos tienen en cuenta estímulos externos como la temperatura, los gradientes de oxígeno y salinidad, la velocidad del agua y los campos eléctricos. Si bien estos estímulos están presentes en la naturaleza, no son los más adecuados en el momento de modelar el comportamiento individual de un pez puesto que dichos estímulos externos pueden ser los que determinen la dirección del movimiento de los peces y no la interacción entre ellos.

El modelo biológico que se ha utilizado para la implementación del simulador es el definido por Huth y Wissel [37]. Este modelo contempla que la organización de los individuos en un banco de peces se mantiene con el transcurso del tiempo sin un líder y no tiene en cuenta estímulos externos. De esta manera, el comportamiento de los individuos y su interacción, es lo que determina la dinámica del grupo.

Los conceptos en los que se han basado los autores del modelo para su desarrollo son los siguientes:

- El movimiento de un pez es influenciado solamente por la posición y orientación de sus vecinos más cercanos.

- La nueva velocidad y ángulo de giro de cada pez, después de un “paso” de tiempo, es determinada por una distribución de probabilidad teniendo en cuenta las influencias aleatorias.

- El movimiento de cada pez modelado está basado en el mismo modelo de comportamiento, por ejemplo, el desplazamiento del banco de peces se basa en el movimiento de los individuos. Dicho movimiento se realiza sin el seguimiento de un líder.

Los patrones de comportamiento básicos modelados por Huth y Wissel son la atracción, la repulsión, la orientación paralela y la búsqueda de vecinos. Cada uno de estos comportamientos responden a cuestiones biológicas vinculadas con el individuo tales como, evitar colisiones, protección y huir de los depredadores.

Además de modelar los patrones de comportamiento mencionados, los autores han demostrado que el comportamiento del banco de peces se asemeja al de uno real si la reacción final de un individuo es el promedio entre los comportamientos resultantes con sus vecinos. De esta forma se consigue una alta cohesión y paralelismo en el movimiento. Si solamente es considerada la interacción con un solo individuo, el movimiento del grupo resultaría confuso.

Modelo biológico

El modelo desarrollado por Huth y Wiessel se basa en los siguientes supuestos:

1. El modelo de comportamiento para cada uno de los peces que forman parte del banco es el mismo. Con esto se garantiza que el grupo se moverá sin la influencia de ningún líder.

2. El movimiento del grupo de peces no está afectado por influencias externas.

3. Influencias aleatorias son tomadas en cuenta para el individuo. De esta forma, la posición y la velocidad de cada pez se basan en variables estocásticas.

4. El movimiento de cada pez está influenciado solamente por sus vecinos cercanos.

5. Se consideran los comportamientos que son decisivos para la organización del banco de peces. Las conductas que no hacen a la formación del cardume no son modeladas puesto que dificultarían la comprensión de los resultados.

El movimiento de los peces se determina por un conjunto básico de patrones de comportamientos que indican cómo se comporta un pez al interactuar con otro individuo solamente.

El conjunto básico de patrones de conductas está constituido por cuatro reacciones:

- Repulsión

- Orientación paralela

- Atracción
- Búsqueda de vecinos

Cada una de estas reacciones depende de la posición en la que se encuentren sus vecinos [4]. En el gráfico de la figura 12 se observan las cuatro zonas correspondientes a cada una de las reacciones. Estas zonas se encuentran delimitadas por tres circunferencias concéntricas y la zona posterior al pez en la que su visión es nula y que se denomina DA (Dead Area). Esta ultima región esta definida por el ángulo ω.

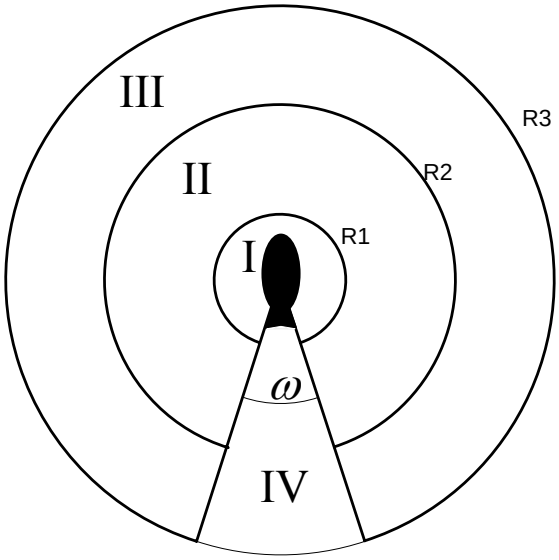


Figura 12: Áreas de influencia

En un banco de peces, la distancia con el vecino más cercano (Nearest Neighbour Distance, NND) varía para cada especie pero tiene un valor promedio que se puede encontrar entre 0,3 a 3 veces el largo del pez (Body Lengths, BL).

Definiendo r como la distancia entre el pez y un vecino, las reacciones en función de las distintas zonas, se pueden expresar como:

- Repulsión, si $r < R_1$ y no se encuentra en DA
- Orientación paralela, si $R_1 < r < R_2$ y no se encuentra en DA
- Atracción, si $R_2 < r < R_3$ y no se encuentra en DA
- Búsqueda de vecinos, si $r > R_3$ y se encuentra en DA

Al igual que la distancia mínima entre vecinos, los valores de los radios son funciones de BL. Valores típicos de los radios son: 0.5BL, 2BL y 5BL para R1, R2, y R3 respectivamente. Elángulo ω tiene como valor típico 30º. En el modelo, los peces se encuentran representados por las coordenadas de su posición $\vec{P_i}$ y por un vector de velocidad $\vec{v_i}$.

La nueva posición y velocidad se calcula en base a las interacciones que tiene el *i-ésimo* pez con los *j-ésimos* peces vecinos. Estas variaciones corresponden a las que se producirían en la realidad al cabo de un incremento de tiempo Δt. Como resultado de la interacción entre un par de individuos, el vector de velocidad $\vec{v_i}$ del *i-ésimo* es rotado un ángulo β_{ij} respecto a su posición previa a la interacción.

El modelo definido por Huth y Wiessel describe la interacción entre los individuos de un banco de peces que se encuentran en un espacio bidimensional. Aunque estas definiciones parecen ser lo suficientemente generales como para ser aplicadas en un espacio tridimensional, esta tarea no es trivial y requiere de la redefinición de diversos conceptos utilizados en las reglas para el espacio bidimensional. A continuación se detallarán las interacciones enumeradas anteriormente y que servirán como base para el desarrollo del modelo de interacciones para un espacio tridimensional.

La orientación paralela es la orientación a la que tienden los bancos de peces durante su desplazamiento. Por lo tanto, existirán ciertas reacciones que tiendan a favorecer a este tipo de movimiento.

Las cuatro reacciones modeladas pueden ser divididas en dos grupos, uno estaría integrado por las reacciones de repulsión, atracción y búsqueda de vecinos que son las reacciones que tienden o favorecen el alineamiento paralelo. El segundo grupo estaría conformado solamente por la reacción de orientación paralela que es la orientación a la que tiende el banco por naturaleza.

- Orientación paralela:

  Si el *j-ésimo* vecino se encuentra entre R_2 y R_3 entonces el ángulo de rotación del vector de velocidad del *i-ésimo* pez será:

$$\beta_{ij} = \angle\left(\vec{v_i}, \vec{v_j}\right) \tag{5}$$

donde $\vec{v}_i$ y $\vec{v}_j$ son los vectores de velocidad del *i-ésimo* y el *j-ésimo* pez respectivamente.

Esta reacción tiende a que el *i-ésimo* pez nade en la misma dirección que el *j-ésimo* (figura 13). De esta forma, si la distancia entre dos individuos es la correspondiente a la orientación paralela, se desplazarán con este tipo de alineamiento.

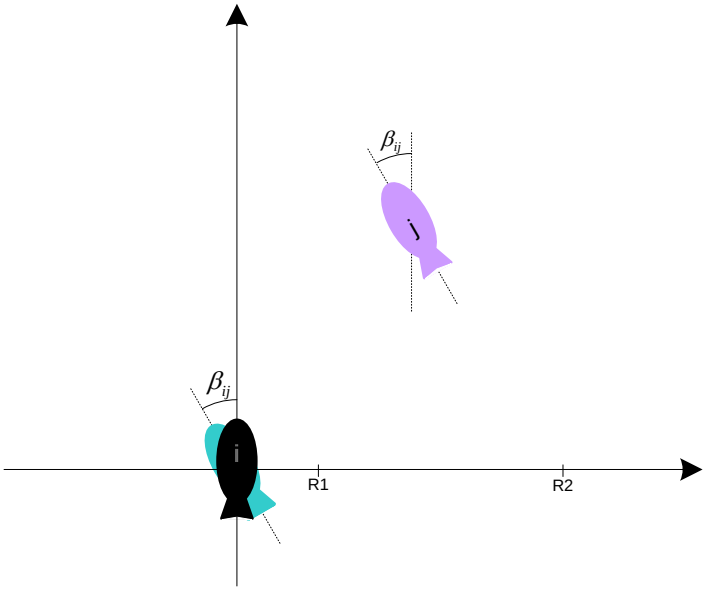


Figura 13: Orientación paralela

- Repulsión:

Durante el desplazamiento del banco de peces, la distancia entre los individuos no se mantiene constante a lo largo del tiempo. Cuando la distancia entre dos peces decrece por debajo de R_1, éstos acostumbran a realizar giros en su movimiento para evitar colisiones.

Este comportamiento es modelado mediante una rotación del vector de velocidad determinado por:

$$\beta_{ij} = \min\left\{\angle\left(\vec{v}_i, \vec{v}_j\right) \pm 90^o\right\} \tag{6}$$

De esta forma, el pez gira para nadar perpendicularmente a la dirección del vector de velocidad de su vecino (Figura 14).

De los dos posibles sentidos marcados por una dirección paralela al vector de velocidad del vecino en un espacio bidimensional, el *i-ésimo* pez escoge la que implique en su rotación, un ángulo menor respecto a su vector velocidad.

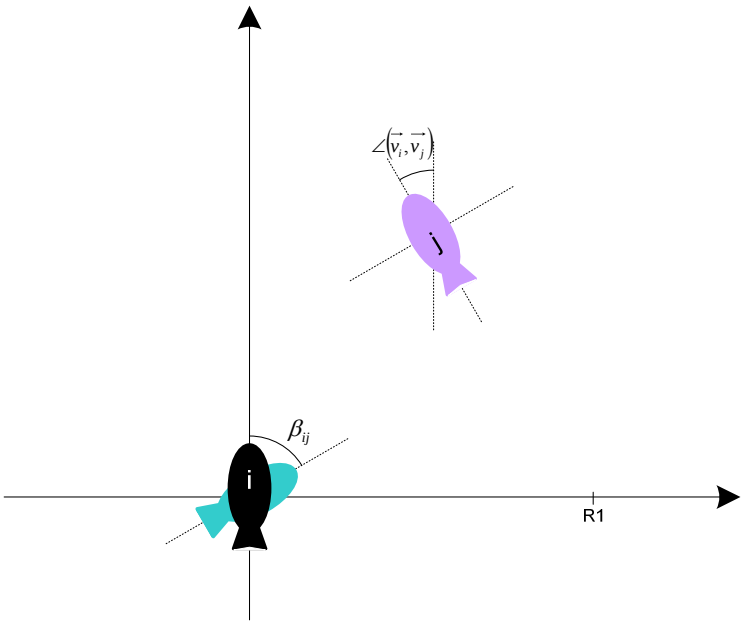


Figura 14: Repulsión

- Atracción:

Cuando la distancia entre un individuo y el pez más próximo a él es mayor que R_3, el individuo reacciona de manera tal que se acerca a su vecino. Esto se debe, entre otros motivos, a un comportamiento biosocial entre individuos de la misma especie. Otro de los motivos por los cuales un pez que se encuentra alejado del grupo tiende a acercarse, es por cuestiones de proteccionismo tratando, de esta manera, evitar ser atacado por los depredadores.

La regla definida por Huth y Wiessel para el caso en el que el vecino se encuentra a una distancia comprendida entre R_2 y R_3 es:

$$\beta_{ij} = \angle\left(\vec{v_i}, \vec{P_j} - \vec{P_i}\right) \tag{7}$$

De esta forma, la reacción del *i-ésimo* individuo es desplazarse en la misma dirección que el vector resultante de la diferencia de $\vec{P_j} - \vec{P_i}$ (figura 15).

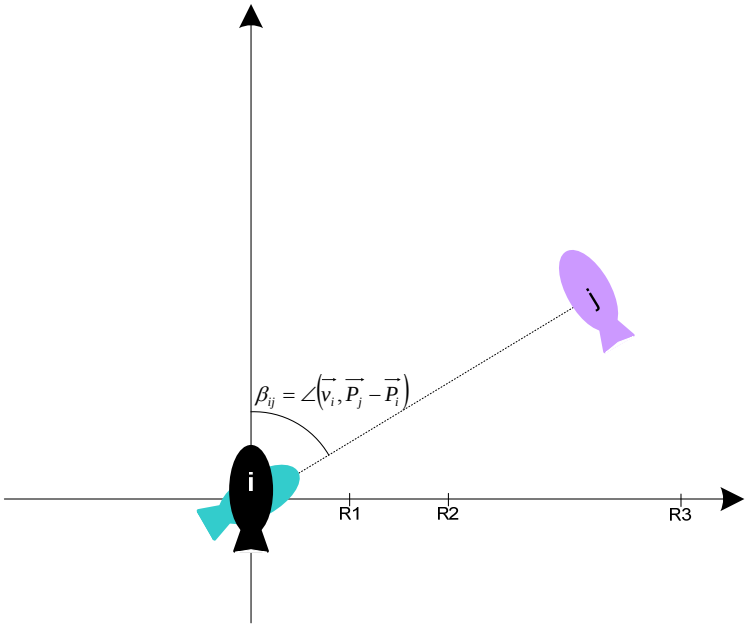


Figura 15: Atracción

- Búsqueda de vecinos:

Por último, la cuarta regla tiene efecto cuando los vecinos se encuentran a una distancia mayor que R_3 o en la zona DA. En este caso, el *i-ésimo* pez comienza la búsqueda de vecinos variando su ángulo en forma aleatoria:

$$\beta_{ij} = chance([-180^o, 180^o]) \tag{8}$$

La determinación de la posición y velocidad de un vecino no es exacta por parte del *i-ésimo* pez, cuenta con cierto grado de incertidumbre. Esto hace que el valor del ángulo de rotación β_{ij}, que se calculó en forma exacta aplicando el modelo sumado a influencias aleatorias, no sea el real. Para contemplar estas incertidumbres e influencias aleatorias, se construye una distribución de probabilidad $p(\alpha_i)$ para el ángulo de rotación β_{ij}.

En la figura 16 se puede observar la distribución de probabilidad $p(\alpha_i)$ para el caso de la interacción con un solo vecino. La media de la función $p(\alpha_i)$ es β_{ij}. De esta forma se contemplan la incertidumbre y las influencias aleatorias que llevan a que el ángulo β_{ij} varíe en torno al valor calculado exacto.

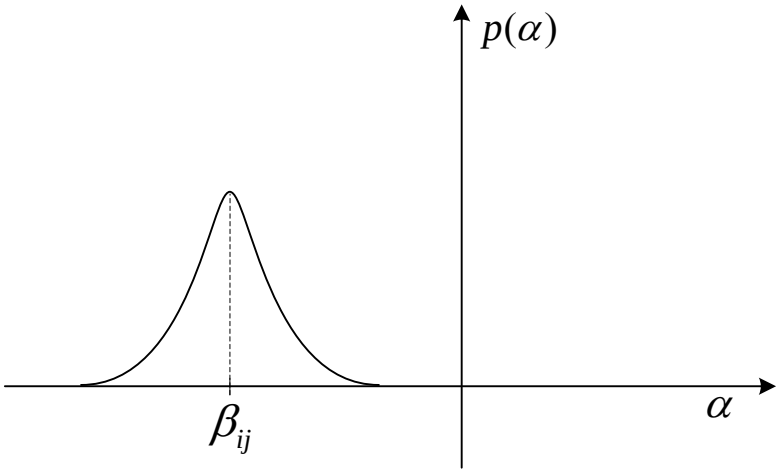


Figura 16: Distribución de probabilidad para el cálculo de β_{ij}

Con $p(\alpha_i)$ un número aleatorio es calculado el cual representa el ángulo de giro α_i escogido por el pez.

$$\alpha_i = chance(p(\alpha_i)) \tag{9}$$

Para simplificar el modelo, Huth y Wissel han considerado a la nueva velocidad (módulo) v_i del *i-ésimo* individuo independiente de los otros peces. La velocidad es calculada al azar tomando una distribución Gamma obtenida en base a datos experimentales [4] (figura 17).

$$v_i = chance(p(v)) \tag{10}$$

$$p(v) = \frac{A^K}{T(K)} e^{(-Av)v^{K-1}} \tag{11}$$

donde $T(K)$ es la función Gamma, v la velocidad, $K = 4$ y $A = 3{,}3$ parámetros.

La distancia desplazada, al igual que los radios de las zonas de interacción de los peces, es medida en BL. La velocidad promedio es de $1{,}2 BL\,\text{sec}^{-1}$.

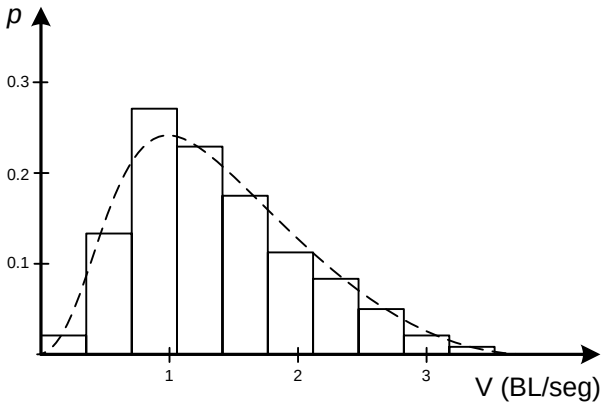


Figura 17: Típica distribución de frecuencia de velocidad en un banco de peces determinado experimentalmente y fijada por una distribución Gamma

Las reacciones analizadas hasta el momento son las que tendría un pez si interactuara solamente con otro. Teniendo en cuenta el cuarto supuesto en el que se basa el modelo de Huth y Wissel que determina que:

"*el movimiento de cada pez es influenciado solamente por los vecinos cercanos*"

se aprecia que aún queda analizar cómo es la rotación final producto de la interacción con los vecinos cercanos.

El ángulo final de rotación es el promedio de las influencias que ejercen cada uno de los vecinos. La distribución final de probabilidad resultante para el ángulo de giro α_i es:

$$p(a_i) = \frac{1}{SD\sqrt{2\pi}} e^{-\frac{(\alpha_i - \overline{\beta_{ij}})^2}{2SD^2}} \tag{12}$$

donde $\overline{\beta_{ij}}$ es el promedio de las influencias de los nb vecinos cercanos y queda definida por:

$$\overline{\beta_{ij}} = \sum_{j=1}^{nb} \frac{\beta_{ij}}{nb} \tag{13}$$

Se puede observar que se tiene una distribución normal en torno al valor del promedio $\overline{\beta_{ij}}$.

A continuación mediante un ejemplo se mostrará la interacción de un pez con otros dos que se encuentran en la zona de orientación paralela.

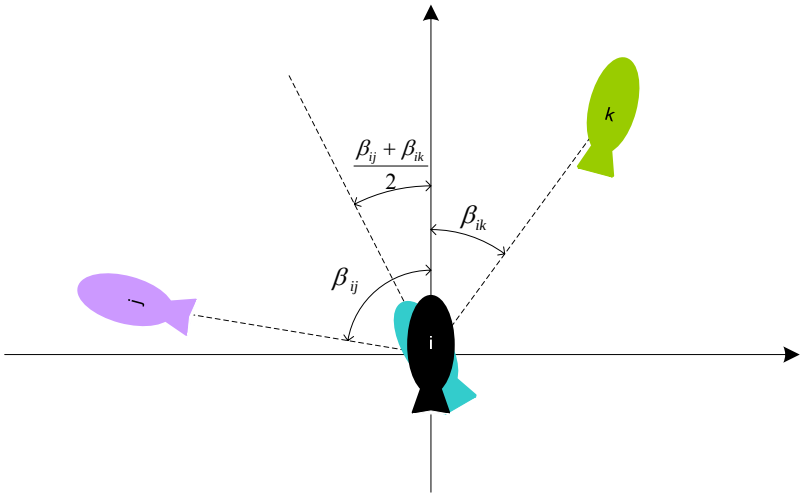


Figura 18: Ejemplo de interacción con dos vecinos

En la figura 18 se pueden apreciar los ángulos de rotación β_{i1} y β_{i2} generados por la interacción del *i-esimo* pez con los individuos 1 y 2 respectivamente. El promedio de ambos está determinado por:

$$\overline{\beta_{ij}} = \sum_{j=1}^{nb} \frac{\beta_{ij}}{nb}; \qquad con \qquad nb = 2 \tag{14}$$

$$\overline{\beta_{ij}} = \sum_{j=1}^{2} \frac{\beta_{ij}}{2} = \frac{\beta_{i1} + \beta_{i2}}{2}$$

La distribución de probabilidad que se muestra en la figura 19 refleja que la distribución normal resultante está centrada en el promedio de las reacciones.

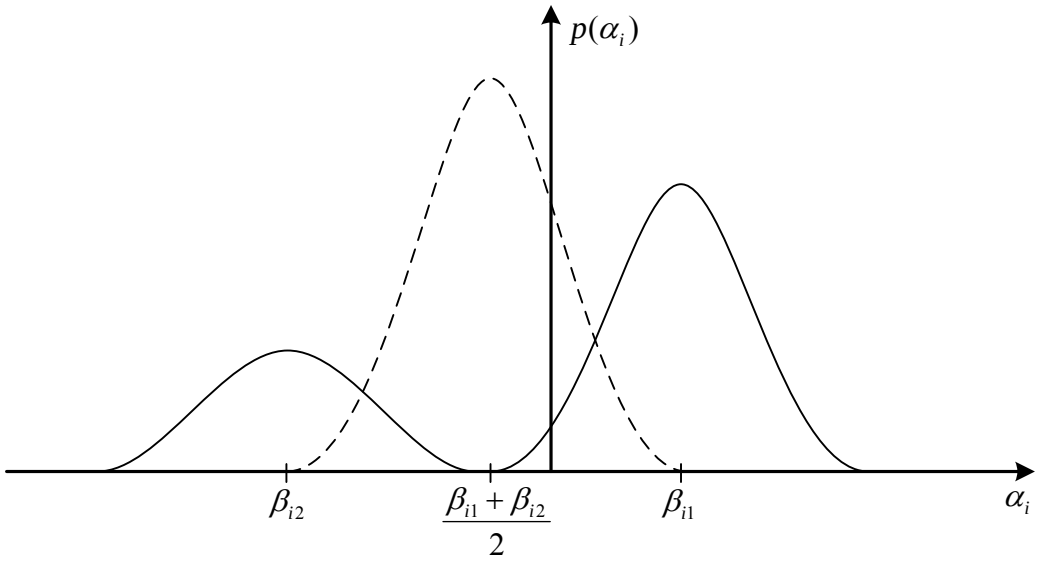


Figura 19: Distribución de probabilidad en la interacción con dos individuos en la zona de orientación paralela

Modelo matemático

El modelo de Huth y Wiessel que se ha comentado a lo largo de este capítulo, describe el comportamiento de los individuos de un banco de peces desde el plano biológico. Si bien este modelo describe todos los comportamientos individuales que determinan el movimiento de un banco de peces, se encuentra limitado al espacio bidimensional.

El modelo matemático desarrollado que será utilizado para la implementación del simulador difiere del realizado por Huth y Wiessel en que es un modelo tridimensional. Con tal fin, se han tenido que replantear diversos conceptos utilizados en el modelo original bidimensional que se detallan a continuación.

En el nuevo modelo, las regiones que determinan el tipo de reacción dejan de estar limitadas por una circunferencia y pasan a ser esferas. Análogamente, la zona de visión nula se transforma en un cono.

El ángulo de rotación $\overline{\beta_{ij}}$ es otro de los conceptos que se ha tenido que adaptar con el paso del modelo bidimensional al espacio tridimensional. Esto se debe a que el ángulo entre dos rectas que no se cortan en el espacio, carece de sentido físico. Un ejemplo de esto es lo que sucedería si dados dos individuos *i* y *j* definidos por:

$$\vec{P_i} = (10,5,5)$$
$$\vec{v_i} = (0.57,0.57,0.57)$$
$$\vec{P_j} = (100,3,9)$$
$$\vec{v_j} = (-0.57,0.57,0.57)$$

se quisiera calcular $\beta_{ij} = \angle\left(\vec{v_i}, \vec{v_j}\right)$ para la reacción de orientación paralela. Este ángulo no existe a menos que $\vec{v_i}$ y $\vec{v_j}$ sean coplanares.

El resultado final de una rotación en $\overline{\beta_{ij}}$ grados, tanto en el plano como en el espacio, implica que el vector de velocidad $\vec{v_i}$ tome una nueva dirección.

Si bien la utilización de ángulos entre rectas en un plano, simplifica el cálculo de la rotación de $\vec{v_i}$, en el espacio se pueden realizar mediante el sistema cartesiano de coordenadas.

En el plano, $\overline{\beta_{ij}}$ es el promedio de la influencia de todos los β_{ij} generados por los vecinos cercanos. En el espacio la nueva velocidad $\vec{v_i}$ se obtiene como la suma de todos los vectores $\vec{v_{ij}}$ generados por la influencia de los *j* vecinos [60].

A continuación se redefinirán las cuatro reglas que rigen el comportamiento de los peces para el espacio tridimensional.

- Orientación paralela:

  Calcula el alineamiento al que por naturaleza tiende el banco de peces. Si el *j-ésimo* individuo se encuentra entre las esferas de radio R_1 y R_2 y fuera de zona de visión nula, el *i-ésimo* pez tomara como nuevo vector de velocidad el del *j-ésimo* individuo. De esta forma, el nuevo vector de velocidad que determinará el desplazamiento paralelo será:

$$\vec{v_{ij}} = \vec{v_j} \tag{15}$$

El concepto de rotación de un ángulo $\beta_{ij} = \angle\left(\vec{v_i}, \vec{v_j}\right)$ entre dos rectas que no se cortan en el espacio y que no forman un plano, carece de sentido físico. Inclusive la expresión $\beta_{ij} = \angle\left(\vec{v_i}, \vec{v_j}\right)$ en el espacio, para dos individuos para los cuales las rectas definidas por sus direcciones no se cortan, el ángulo β_{ij} no queda definido.

- Repulsión:

  Mediante esta regla, se expresa el comportamiento de los peces para evitar colisiones con los vecinos que se encuentran muy próximos. Este tipo de reacción tiene lugar con los individuos que se hallan dentro de la esfera de radio R_1 y fuera de DA.

  La forma en la que el *i-ésimo* pez evita colisionar con el *j-ésimo* es rotando su vector velocidad de forma tal que su desplazamiento sea perpendicular al desplazamiento del otro pez.

  En el plano esta reacción quedaba definida por (6):

$$\beta_{ij} = \min\left\{\angle\left(\vec{v_i}, \vec{v_j}\right) \pm 90^{\circ}\right\}$$

En el espacio, hallar una dirección perpendicular a la dirección del *j-ésimo* individuo no es trivial. Cumplir con esta condición no es simple como en el plano donde solo existen dos posibilidades.

El problema radica en los infinitos vectores que determinan una dirección perpendicular a la dirección del *j-ésimo* pez. Los infinitos vectores contenidos en los infinitos planos perpendiculares al vector $\vec{v}_j$ cumplen con la condición de perpendicularidad.

De la definición hecha por Huth y Wiessel de la reacción de repulsión en el plano ($\beta_{ij} = \min\left\{\angle\left(\vec{v}_i, \vec{v}_j\right) \pm 90^o\right\}$), se extrae que el ángulo por el que se opta, es el que implica un ángulo menor de rotación del vector $\vec{v}_i$. La aplicación de este concepto al espacio tridimensional, requiere de un análisis vectorial previo.

Si bien los vectores $\vec{v}_i$ y $\vec{v}_j$ son los que determinan la dirección y sentido de los peces a los que hacen referencia, y que se encuentran situados en puntos distintos del espacio, ambos vectores pueden ser analizados ubicándolos en el origen del sistema de coordenadas.

Como se ha dicho anteriormente, existen infinitos planos que contienen los infinitos vectores perpendiculares al desplazamiento del *i-ésimo* pez. De todos estos planos, el que se utilizará para el cálculo de la reacción, es el que cumpla con las condiciones de contener al origen y ser perpendicular al desplazamiento del *j-ésimo* individuo.

Una vez encontrado este plano, se debe hallar el vector $\vec{v}_{ij}$ que cumpla con la condición de pertenecer al plano formado por $\vec{v}_i$ y $\vec{v}_j$, que sea a perpendicular a $\vec{v}_j$ y que se encuentre contenido en el plano perpendicular a $\vec{v}_j$. Lo dicho anteriormente implica que el ángulo de rotación de $\vec{v}_i$ para alcanzar al nueva posición determinada por $\vec{v}_{ij}$ sea el menor.

Estas condiciones pueden resumirse de la siguiente manera:

- $\vec{v}_{ij} \perp \vec{v}_j$ y $\vec{v}_{ij}$ pertenece al plano formado por los vectores $\vec{v}_i$ y $\vec{v}_j$

En la figura 20 se observa un ejemplo sencillo de lo explicado anteriormente en el cual el vector $\vec{v}_j$ esta definido por la terna $(v_{x_j},0,0)$ y el $\vec{v}_i$ por $(v_{x_i},v_{y_i},0)$ De esta forma, el plano perpendicular a $\vec{v}_j$ que contiene a todos los vectores perpendiculares a éste y pasa por el origen, es el plano YZ. El plano formado por los vectores $\vec{v}_i$ y $\vec{v}_j$ que contiene al vector $\vec{v}_{ij}$ es el plano XY. Como se puede apreciar, el $\vec{v}_{ij}$ es el vector que cumple con las condiciones mencionadas antes.

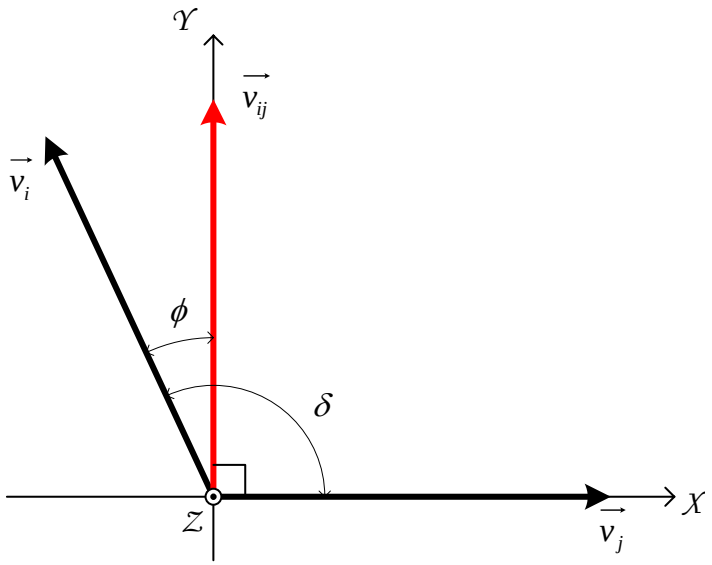


Figura 20: Ejemplo de reacción de repulsión

En forma general las dos condiciones necesarias para encontrar $\vec{v}_{ij}$ pueden expresarse matemáticamente de la siguiente forma:

teniendo en cuenta que el producto escalar de dos vectores $\vec{A}$ y $\vec{B}$ queda definido por:

$\vec{A} \bullet \vec{B} = |A| \cdot |B| \cos\varphi$, donde φ es el ángulo entre los vectores

y si $\varphi = \frac{\pi}{2}$ (condición de perpendicularidad):

$$\vec{A} \bullet \vec{B} = 0$$

se puede aplicar este caso para cumplir con la primer condición de perpendicularidad entre $\vec{v}_{ij}$ y $\vec{v}_j$:

$$\overrightarrow{v_{ij}} \bullet \overrightarrow{v_j} = 0 \tag{16}$$

Definiendo:

$$\overrightarrow{v_i} = \left(v_{x_i}, v_{y_i}, v_{z_i}\right),\ \overrightarrow{v_j} = \left(v_{x_j}, v_{y_j}, v_{z_j}\right),\ \text{y}\ \overrightarrow{v_{ij}} = \left(v_{x_{ij}}, v_{y_{ij}}, v_{z_{ij}}\right) \tag{17}$$

se tiene que el producto escalar $\overrightarrow{v_{ij}} \bullet \overrightarrow{v_j}$ es igual a:

$$v_{x_{ij}} \cdot v_{x_j} + v_{y_{ij}} \cdot v_{y_j} + v_{z_{ij}} \cdot v_{z_j} = 0 \tag{18}$$

y de esta ecuación se obtienen los infinitos vectores perpendiculares a $\overrightarrow{v_j}$.

Considerando que $\overrightarrow{v_{ij}}$ es coplanar con $\overrightarrow{v_i}$ y $\overrightarrow{v_j}$ se llega a que $\overrightarrow{v_{ij}}$ puede ser expresado como combinación lineal de los vectores generadores del plano:

$$\overrightarrow{v_{ij}} = K_1\overrightarrow{v_i} + K_2\overrightarrow{v_j} \tag{19}$$

de esta forma se tiene el siguiente sistema de ecuaciones:

$$\begin{cases} v_{x_{ij}} \cdot v_{x_j} + v_{y_{ij}} \cdot v_{y_j} + v_{z_{ij}} \cdot v_{z_j} = 0 \\ \overrightarrow{v_{ij}} = K_1\overrightarrow{v_i} + K_2\overrightarrow{v_j} \end{cases} \tag{20}$$

Tomando $K_1 = 1$ y reemplazando la segunda ecuación en la primera se llega a:

$$K_2 = -\frac{v_{x_i} \cdot v_{x_j} + v_{y_i} \cdot v_{y_j} + v_{z_i} \cdot v_{z_j}}{v_{x_j}^2 + v_{y_j}^2 + v_{z_j}^2} \tag{21}$$

De esta forma se puede conocer el valor de las componentes de $\overrightarrow{v_{ij}}$ mediante simples operaciones algebraicas.

$$\begin{cases} x_{ij} = x_i - \dfrac{v_{x_i} \cdot v_{x_j} + v_{y_i} \cdot v_{y_j} + v_{z_i} \cdot v_{z_j}}{v_{x_j}^2 + v_{y_j}^2 + v_{z_j}^2} \cdot x_j \\ y_{ij} = y_i - \dfrac{v_{x_i} \cdot v_{x_j} + v_{y_i} \cdot v_{y_j} + v_{z_i} \cdot v_{z_j}}{v_{x_j}^2 + v_{y_j}^2 + v_{z_j}^2} \cdot y_j \\ z_{ij} = z_i - \dfrac{v_{x_i} \cdot v_{x_j} + v_{y_i} \cdot v_{y_j} + v_{z_i} \cdot v_{z_j}}{v_{x_j}^2 + v_{y_j}^2 + v_{z_j}^2} \cdot z_j \end{cases} \quad (22)$$

- Atracción:

La finalidad de esta regla es mantener unido al grupo mediante la atracción de los peces que se encuentran más alejados del banco de peces. Esto responde a un comportamiento biosocial entre individuos de la misma especie.

En forma análoga con las otras reglas, los vecinos que generan este tipo de reacción son los que se encuentran entre las esferas de radio R_2 y R_3 y fuera de la zona de visión nula.

La forma en la que un individuo es atraído por otro, es por medio de un desplazamiento determinado por el vector definido por la diferencia ente la posición del *j-ésimo* y el *i-ésimo* individuo.

El vector velocidad para la interacción entre el *i-ésimo* y el *j-ésimo* pez es:

$$\overrightarrow{v_{ij}} = \overrightarrow{P_j} - \overrightarrow{P_i} \quad (23)$$

siendo $\overrightarrow{P_j}$ y $\overrightarrow{P_i}$ las coordenadas de los peces *j-ésimo* e *i-ésimo*.

- Búsqueda de vecinos:

La cuarta y última de las reacciones se produce cuando la distancia mínima entre el *i-ésimo* y el *j-ésimo* pez es mayor que R_3 o se encuentra en la zona de visión nula. En esta situación, los peces realizan movimientos aleatorios para encontrar otros peces y formar un banco. Durante esta búsqueda también puede suceder que el *i-ésimo* pez se encuentre cerca de un banco de peces. Si esto sucede, el pez será atraído al grupo por una reacción de atracción en cuanto su distancia al banco de peces sea menor que R_3.

En el modelo definido por Huth y Wiessel, la búsqueda de vecinos está determinada por un movimiento de β_{ij} dado por:

$$\beta_{ij} = chance([-180º, 180º]) \tag{24}$$

Esto implicaría que las coordenadas cartesianas del vector velocidad $\overrightarrow{v_{ij}}$ bidimensional acorde al valor que tome β_{ij}.

Para el modelo matemático tridimensional en coordenadas cartesianas se tiene:

$$\begin{aligned} \overrightarrow{v_{ij}} &= \left(v_{x_{ij}}, v_{y_{ij}}, v_{z_{ij}}\right) \\ v_{ij_x} &= chance([-1,1]) \\ v_{ij_y} &= chance([-1,1]) \\ v_{ij_z} &= chance([-1,1]) \end{aligned} \tag{25}$$

De esta forma se consigue un movimiento aleatorio en el espacio equivalente al conseguido en el plano por medio de la variación de β_{ij}.

Modelo computacional

Una vez obtenido el modelo matemático del sistema, el próximo paso para la implementación del simulador, es el desarrollo de un modelo computacional. Para lograr este cometido, no solo se deben tener en cuenta temas relacionados con la biología y matemática, sino también con aspectos vinculados a la implementación en programas de cómputo.

Muchos de los conceptos desarrollados no tuvieron en cuenta las modificaciones que se deberían realizar o considerar para poder llevar a cabo un modelo que pueda ser implementado en un simulador. Dichas consideraciones giran entorno a la diferencia con la que se trata el tiempo y el espacio en el modelo matemático y en el computacional. En el primero se consideran los cambios espaciales y temporales en forma continua mientras que en el segundo se hace en forma discreta.

La discretización del tiempo implica que las posiciones y velocidades se calcularán cada determinado intervalo de tiempo (paso, T_p). Si el paso es considerable, la exactitud en el cálculo decrece ya que entre dos instancias de cálculo consecutivas, no se han tenido en cuenta las posibles interacciones que hubieran existido entre los individuos. Por otro lado, si

el tiempo asignado a un paso es demasiado pequeño, el error acarreado por la influencia de la discretización del espacio puede ser considerable.

El vector de velocidad en un sistema continuo está definida por:

$$\overrightarrow{velocidad} = \frac{d\vec{r}}{dt} \text{; con } \vec{r} \text{ el vector de posición} \tag{26}$$

En el modelo computacional, se considerará que la posición sólo cambiará en instantes de tiempo discretos, entonces se puede redefinir la velocidad como:

$$\frac{\Delta\vec{r}}{\Delta t} \tag{27}$$

Además de estas consideraciones, en el modelo se ha tomado al módulo de velocidad constante a lo largo de toda la simulación. Esa consideración no implica que la velocidad sea constante, puesto que cuando un pez rota su vector de velocidad, se producirá una aceleración tangencial (aunque sea de módulo constante). Esta aceleración tangencial es consecuencia de una variación temporal en la velocidad tangencial.

El otro aspecto que se debe incluir en el modelo matemático para la obtención del modelo computacional, es la discretización espacial.

En la presente tesis, el espacio ha sido discretizado en enteros. Cada paso espacial corresponde a un entero de longitud 1BL. De esta forma, se agrega un grado de incertidumbre a los valores exactos obtenidos en la simulación ya que estos son expresados durante el cálculo en punto flotante.

En el ejemplo de la figura 21 se muestra gráficamente lo expresado en el párrafo anterior para el caso de un desplazamiento en dos dimensiones. La nueva posición P_{i+1} obtenida en base a la posición previa P_i, se encuentra entre cuatro posibles puntos definidos en el plano por la discretización. Al utilizarse números enteros para la representación del espacio, P_{i+1} (expresado en flotante) no queda definida. Por tal motivo, se efectuará un desplazamiento del punto exacto hacia alguno de los cuatro puntos más cercanos que se encuentran marcados por la discretización del espacio. De esta manera, se determina la nueva posición P'_{i+1} para el tiempo $i+1$.

En el espacio tridimensional se puede hacer un planteo análogo con la diferencia de que la nueva coordenada estará ubicada entre ocho puntos.

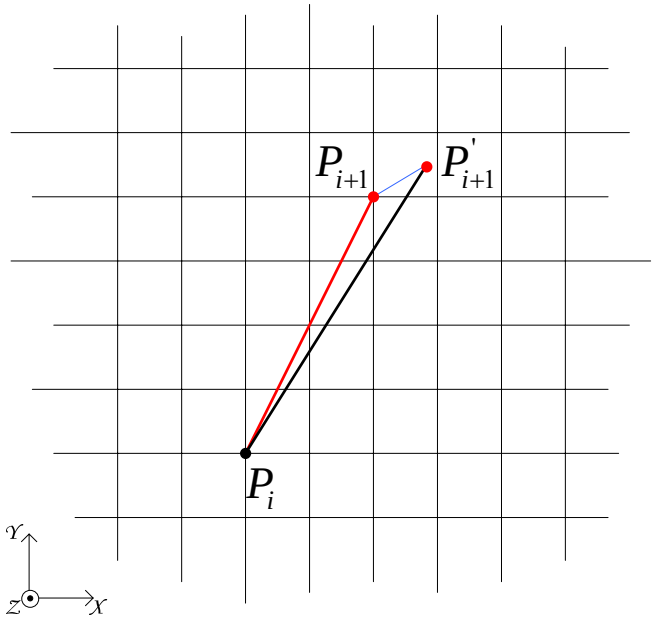


Figura 21: Discretización del espacio

La diferencia entre el valor exacto de la posición y el que se escogerá para definir el nuevo punto, generará un error. Suponiendo un espacio tridimensional y un módulo de velocidad de ***v***, el error cometido en el peor de los casos es cuando el valor exacto se encuentra en un vértice del cubo y el punto escogido se encuentra en el otro extremo de la diagonal mayor. La distancia entre P_i y P_{i+1}, estará determinada por el producto de ***v*** con el paso de tiempo T_p.

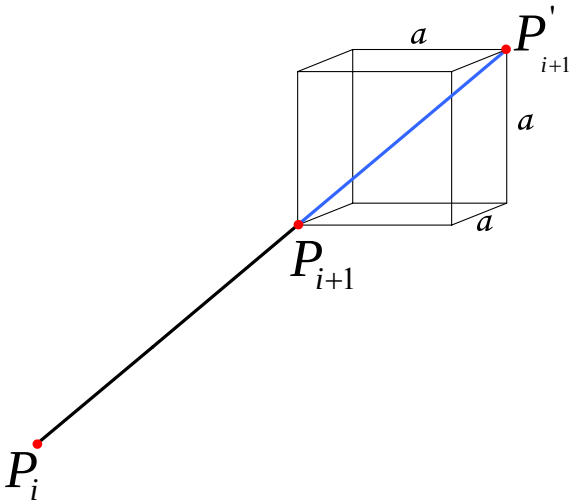


Figura 22: Efecto de la discretización espacial

Por lo tanto el error cometido para una discretización espacial de ***a*** será:

$$\pm\sqrt{3}\,\boldsymbol{a} \tag{28}$$

Como puede observarse, el error es independiente de la velocidad y depende únicamente del paso de discretización espacial con lo cual queda de manifiesto que la elección de la velocidad debe ser realizada teniendo en cuenta el valor ***a*** y T_p de la discretización para no cometer un error elevado.

La distancia entre el punto P_i y P'_{i+1} estará acotada por la siguiente fórmula:

$$d(P_i, P_{i+1}) = vT_p \pm \sqrt{3}\,a \tag{29}$$

Esta variación en la posición exacta podría interpretarse asimismo como una variación en el módulo y/u orientación del vector de velocidad que producen que el punto P_{i+1} se ubique en la posición P'_{i+1}.

Asumiendo que la variación entre P_{i+1} y P'_{i+1} se deben a una variación del vector de velocidad, estos se pueden utilizar para satisfacer ciertas especificaciones del modelo biológico.

En el apartado de modelo biológico, se ha dicho que tanto la posición como la dirección y sentido de los *j-ésimos* vecinos no pueden ser determinados en forma exacta en la realidad por el *i-ésimo* pez. También existen influencias aleatorias que llevan a que la posición y velocidad calculada sean modeladas como una distribución de probabilidad gaussiana con media en el valor exacto.

Si bien la ubicación de la nueva posición en alguna de las esquinas del cubo genera un error con respecto al valor exacto, en el modelo computacional, se lo considerará como el efecto de la incertidumbre en la determinación de la posición de los *j-ésimos* vecinos y las influencias aleatorias.

El desplazamiento a las esquinas del cubo es el mínimo movimiento que se puede obtener teniendo en cuenta la discretización espacial. La elección de alguno de los ocho vértices se hace en forma aleatoria. De esta manera, se tiene un desplazamiento aleatorio equivalente al que se obtendría si se aplicara una distribución de probabilidad normal en torno al valor medio representado por la nueva posición.

La implementación de las reglas de comportamiento en el modelo matemático es, en su gran mayoría, directa. El paralelismo, la atracción y la búsqueda de vecinos no muestran inconvenientes puesto que en el primer caso el resultado es la reacción resultante de igualar el vector de velocidad del *i-ésimo* pez con el del *j-ésimo*. La atracción consiste en tomar el vector diferencia de posiciones como el nuevo vector de velocidad y la búsqueda de vecino tomar cualquier vector de velocidad nuevo.

La repulsión presenta ciertas características que tornan el paso del modelo matemático al computacional no sea directo. A continuación se describen los casos puntuales en los que no puede realizarse el paso comentado anteriormente.

- Caso 1: Velocidades iguales e individuos no colineales.

  Para esta situación, los individuo *i-ésimo* y *j-ésimo* tienen el mismo vector de velocidad y su posición no es colineal (figura 23). El problema que presenta esta situación es que no se puede hallar un vector perpendicular a $\vec{v_j}$, que esté contenido en el plano generado por los vectores $\vec{v_i}$ y $\vec{v_j}$ y que sea combinación lineal de estos.

  Una forma para determinar si dos peces con el mismo vector de velocidad no son colineales es mediante el producto escalar entre alguno de los vectores de velocidad y el vector definido por la diferencia de posición $\vec{D} = \vec{P_j} - \vec{P_i}$ y analizando el valor del coseno del ángulo formado entre ellos:

$$\vec{v_i} \bullet \vec{D} = \left|\vec{v_i}\right| \cdot \left|\vec{D}\right| \cos\left(\angle\left(\vec{v_i}, \vec{D}\right)\right) \tag{30}$$

  Puesto que $\vec{v_i}$ se encuentra normalizado y haciendo lo mismo con $\vec{D}$, el resultado del producto escalar será:

$$\begin{cases} 1 \, o \, -1 & \textit{si son colineales} \\ \text{otro valor} & \textit{si no son colineales} \end{cases}$$

  Puesto que un vector perpendicular a $\vec{v_j}$ también lo es de $\vec{v_i}$ se hallará el nuevo vector $\vec{v_{ij}}$ con las siguientes condiciones:

- $\overrightarrow{v_{ij}}$ es combinación lineal de los vectores $\overrightarrow{D} = \overrightarrow{P_j} - \overrightarrow{P_i}$ y $\overrightarrow{v_i}$
- $\overrightarrow{v_{ij}} \perp \overrightarrow{v_i}$

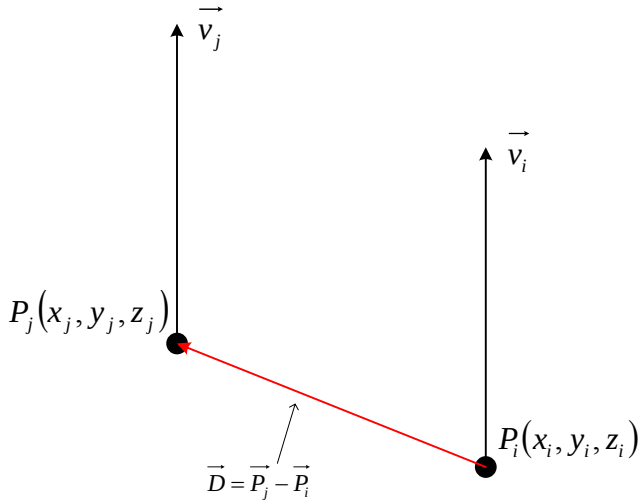


Figura 23: Caso 1, velocidades iguales e individuos no colineales

Aplicando estas dos condiciones y operando matemáticamente, se llega a que las componentes del vector $\overrightarrow{v_{ij}}$ quedan determinadas por:

$$\begin{cases} v_{x_{ij}} = -\dfrac{v_{x_i}(x_j - x_i) + v_{v_i}(y_j - y_i) + v_{z_i}(z_j - z_i)}{{v_{x_i}}^2 + {v_{y_i}}^2 + {v_{z_i}}^2} v_{x_i} + v_{x_j} \\ v_{y_{ij}} = -\dfrac{v_{x_i}(x_j - x_i) + v_{v_i}(y_j - y_i) + v_{z_i}(z_j - z_i)}{{v_{x_i}}^2 + {v_{y_i}}^2 + {v_{z_i}}^2} v_{y_i} + v_{y_j} \\ v_{z_{ij}} = -\dfrac{v_{x_i}(x_j - x_i) + v_{v_i}(y_j - y_i) + v_{z_i}(z_j - z_i)}{{v_{x_i}}^2 + {v_{y_i}}^2 + {v_{z_i}}^2} v_{z_i} + v_{z_j} \end{cases} \quad (31)$$

- Caso 2: Velocidades iguales e individuos colineales.

Al igual que en el caso 1, los vectores de velocidad no pueden generar un único plano en el cual esté contenido el nuevo vector de velocidad $\overrightarrow{v_{ij}}$. Al ser colineales, el producto escalar:

$$\overrightarrow{v_i} \bullet \overrightarrow{D} = \left|\overrightarrow{v_i}\right| \cdot \left|\overrightarrow{D}\right| \cos\left(\angle\left(\overrightarrow{v_i}, \overrightarrow{D}\right)\right) \quad (32)$$

es 1 o -1 lo cual determina la condición de colineal.

En general, se puede expresar la condición de perpendicularidad entre $\overrightarrow{v_{ij}}$ y $\overrightarrow{v_j}$ mediante el producto escalar de la siguiente forma:

$$v_{x_{ij}} v_{x_j} + v_{y_{ij}} v_{y_j} + v_{z_{ij}} v_{z_j} = 0 \tag{33}$$

Puesto que la condición de que $\overrightarrow{v_{ij}}$ pertenezca al plano formado por $\overrightarrow{v_i}$ y $\overrightarrow{v_j}$ no se puede satisfacer, se tomarán aleatoreamente alguna de las seis posibilidades que se detallan a continuación que generan un vector perpendicular a $\overrightarrow{v_j}$ (y también a $\overrightarrow{v_i}$ por ser iguales):

- Posibilidad 1 y 2:

Forzando $v_{x_{ij}} = 0$, entonces se tiene:

$$v_{y_{ij}} v_{y_j} + v_{z_{ij}} v_{z_j} = 0 \tag{34}$$

Y de aquí se tienen las dos posibilidades que son:

$$\begin{cases} v_{y_{ij}} = v_{z_j} \\ v_{z_{ij}} = -v_{y_j} \end{cases} \qquad \begin{cases} v_{y_{ij}} = -v_{z_j} \\ v_{z_{ij}} = v_{y_j} \end{cases} \tag{35}$$

- Posibilidad 3 y 4:

Haciendo $v_{y_{ij}} = 0$, se tiene:

$$v_{x_{ij}} v_{x_j} + v_{z_{ij}} v_{z_j} = 0 \tag{36}$$

Entonces las dos variantes son:

$$\begin{cases} v_{x_{ij}} = v_{z_j} \\ v_{z_{ij}} = -v_{x_j} \end{cases} \qquad \begin{cases} v_{x_{ij}} = -v_{z_j} \\ v_{z_{ij}} = v_{x_j} \end{cases} \tag{37}$$

- Posibilidad 5 y 6:

Con $v_{z_{ij}} = 0$:

$$v_{x_{ij}} v_{x_j} + v_{y_{ij}} v_{y_j} = 0 \tag{38}$$

Las posibilidades son:

$$\begin{cases} v_{x_{ij}} = v_{y_j} \\ v_{y_{ij}} = -v_{x_j} \end{cases} \qquad \begin{cases} v_{x_{ij}} = -v_{y_j} \\ v_{y_{ij}} = v_{x_j} \end{cases} \tag{39}$$

Por último, dentro de las particularidades que se pueden presentar en el modelo computacional, está la posibilidad de que las nuevas posiciones de varios individuos coincidan. Esta situación no puede ser admitida.

Una vez calculadas todas las nuevas posiciones, se verifica que no existan coincidencias. De suceder esto, se modifican las posiciones en incrementos de un paso de discretización espacial hasta que las posiciones sean únicas.

Capítulo 4
Simulador

4.1. Introducción

El modelo matemático descrito en el capítulo anterior ha sido implementado en un simulador distribuido puesto que la complejidad que presenta el modelo hace que la simulación serie consuma una gran cantidad de tiempo.

En este capítulo se desarrolla la forma en la que se implementó el modelo matemático Fish School y se detalla el algoritmo de simulación. Del estudio detallado de este último, se ha obtenido un modelo analítico del simulador que permite predecir tres datos importantes:

- Tiempo de simulación en función de los procesadores y la cantidad individuos. $T_i=f(P,N)$
- Recursos (procesadores) necesarios para simular un determinado número de peces en un tiempo establecido. $P=f(N,T_i)$
- Cantidad de individuos que se pueden simulara utilizando cierta cantidad de procesadores y de tiempo. $N=f(P,T_i)$

Tomando como base el trabajo llevado a cabo, la experiencia adquirida y la investigación realizada, se presenta una alternativa a la simulación distribuida de IoM que permite ampliar el espectro de simulado a más individuos, grupos y espacio de simulación.

4.2. Implementación del simulador

La implementación del modelo matemático se ha realizado en un simulador distribuido. Como se ha mencionado en el capítulo 2, existen dos tipos de técnicas para la simulación distribuida: la optimista y la conservativa. Experiencias previas realizadas en simulación distribuida con el modelo Fish School [72] [54] han demostrado que la utilización del método optimista presenta inconvenientes que no lo hacen apto para este tipo de modelos. El simulador desarrollado utiliza el método conservativo en el cual las listas de eventos es descentralizada.

El tipo de distribución realizada en el modelo de simulación, es una distribución del espacio de simulación (nivel de componentes, ver capítulo 2). El espacio está dividido en bloques (slices) y la simulación se realiza en el espacio que le corresponde a cada uno de los bloques es llevada a cabo por un procesador del sistema. Cada uno de estos bloques corresponde a un proceso lógico (LP).

El simulador se basa en dos tipos de procesos, el proceso inicial (Father) y los *LPs*. En la figura 24 puede observarse la arquitectura del simulador y los dos tipos procesos que lo conforman. El *Father* es el encargado de la inicialización de los datos en los distintos *LPs* y está preparado para que, en futuras extensiones del código, pueda cumplir diversas funciones como, por ejemplo, monitorización centralizada. Una vez finalizada la etapa de inicialización, los *LPs* son los únicos procesos que se mantienen durante el transcurso de la simulación.

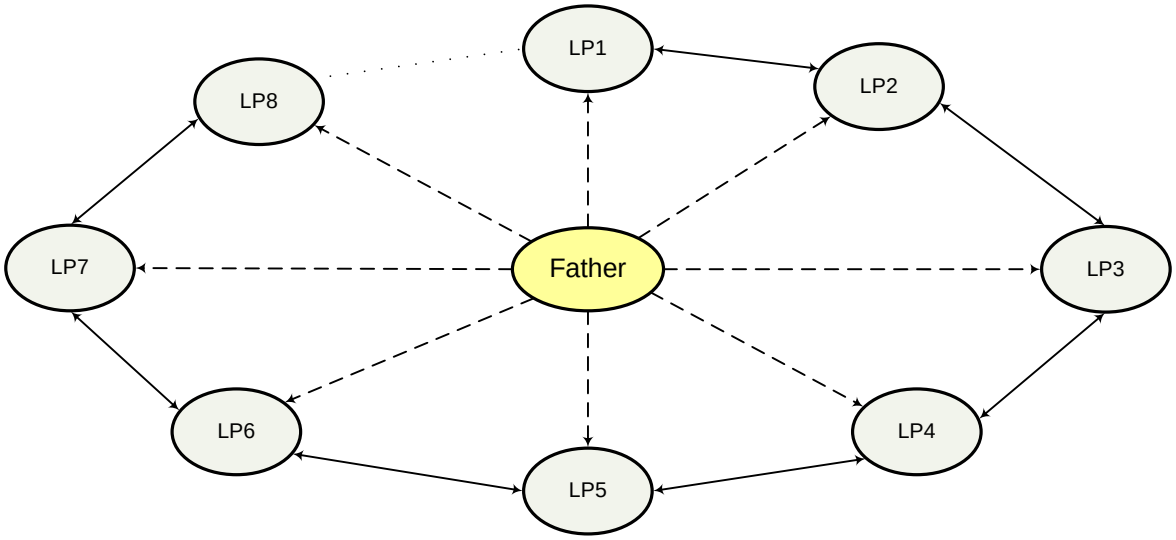


Figura 24: Arquitectura del simulador distribuido

La distribución de los individuos en los diversos *LPs* se puede realizar de dos formas: con ficheros externos o por medio de una distribución aleatoria a lo largo de todo el espacio de simulación. En la inicialización con ficheros externos, es el usuario quien determina la posición y velocidad inicial de cada uno de los individuos del sistema. La otra forma de inicialización asegura que cada uno de los *LPs* disponga de la misma cantidad de individuos en el inicio de la simulación. La distribución se hace de manera tal que la ubicación de los peces sea uniforme en todo el espacio del simulador.

El espacio de simulación es un paralelepípedo conformado por una concatenación de paralelepípedos menores que representan la sección del espacio a simular por cada *LP* (figura 25). La comunicación se realiza entre *LPs* vecinos, es decir, el LP_i se comunica únicamente con el LP_{i-1} y el LP_{i+1}. La comunicación entre el LP_1 y LP_n se produce solamente cuando la nueva posición de un individuo implica atravesar la cara izquierda o derecha del LP_1 o LP_n respectivamente. Este efecto de frontera tiene por objetivo mantener constante el número de individuos y evitar efectos de borde o discontinuidades en el modelo (figura 25). Otra situación que puede presentarse es que la nueva posición se halle fuera del *LP* y no pertenezca a ningún *LP* en forma vertical. Si esto sucede, el pez es incorporado nuevamente por la cara opuesta a la que ha salido. De esta forma, la densidad de individuos en el espacio de simulación se mantiene constante a lo largo de toda la simulación.

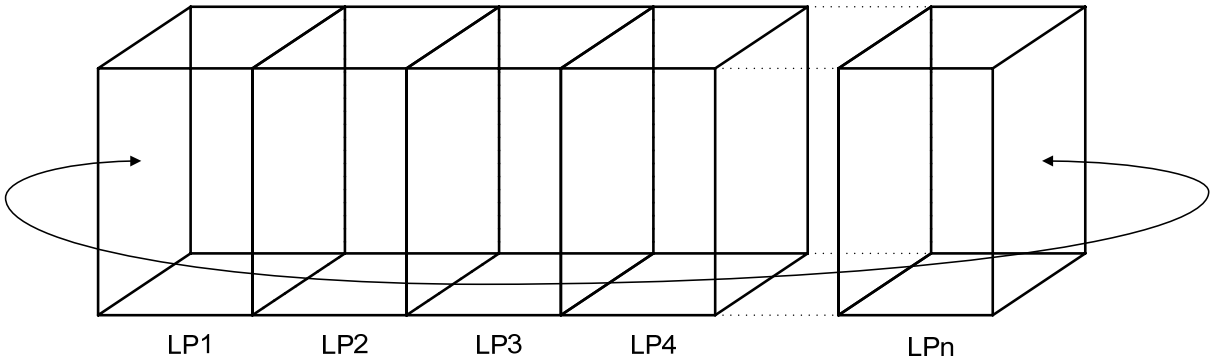


Figura 25: Distribución del espacio de simulación en procesos lógicos

4.3. Funcionamiento y algoritmo de simulación

En el sistema, cada individuo es representado por dos ternas y un número que sirve como identificador único. La primera terna indica las coordenadas de su posición en el espacio de simulación. La segunda contiene las componentes del vector velocidad que determinan en que dirección y sentido se está desplazando el pez.

Por cada iteración, en la simulación se calculan las nuevas posiciones y velocidades de los individuos. El principio de funcionamiento del simulador, en lo concerniente al modelo matemático, se basa en los siguientes pasos, que se realizan en cada iteración y para cada individuo:

- Búsqueda de vecinos potenciales. Estos vecinos son los que se hallan dentro de la esfera determinada por el radio mayor de influencia.

- De los vecinos potenciales hallados, se seleccionan los cuatro que se encuentren más cerca de la línea de visión directa. En el caso de que los vecinos potenciales sean menos de cuatro, no se realiza la selección y se escogen a todos ellos.

- De acuerdo a la posición de los cuatro vecinos se halla la influencia de cada uno de ellos sobre el individuo sobre el que se está calculando su nueva posición (*i-ésimo* individuo). La influencia resultante de la interacción con estos vecinos está basada en la influencia individual de cada vecino como se explicó en el capítulo anterior.

- Una vez obtenida la influencia total, se calcula la nueva terna de velocidad y a continuación la posición nueva.

Si esta simulación fuera realizada en un simulador serie, en el cual todos los individuos se encontraran en el mismo espacio de simulación, la búsqueda de vecinos potenciales (individuos que se encuentran a una distancia menor que el mayor radio de influencia) se realizaría mediante el cálculo de la distancia entre el *i-ésimo* pez y cada uno de los restantes individuos. Como el espacio de simulación está distribuido en los distintos *LPs*, la búsqueda de vecinos potenciales presenta ciertas características. Estas lo diferencian del caso en el que el modelo sea implementado en un simulador serie y serán comentadas a continuación.

Durante la selección de los vecinos potenciales puede suceder que el individuo se encuentre a una distancia de la frontera del bloque contiguo menor que el mayor radio de influencia, (figura 26). Esta situación implica que existe la posibilidad de que alguno de los vecinos potenciales se encuentre en la región vecina, próximo a la frontera y, por consiguiente, sea un vecino potencial.

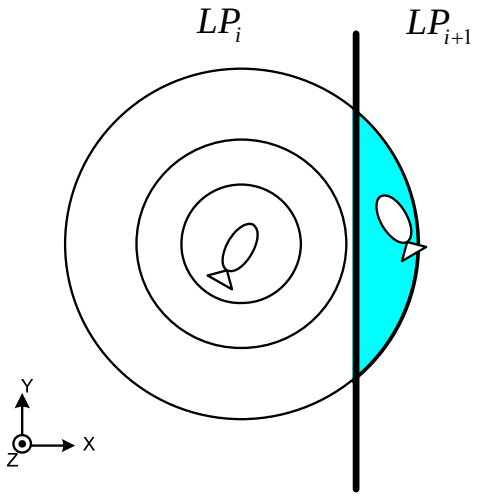


Figura 26: Individuo cerca de la frontera

Puesto que cada bloque contiene solamente la información de las posiciones y velocidades de sus elementos, cuando un individuo presenta las condiciones mencionadas anteriormente el LP_i realizará una consulta al LP_{i+1} (bloque vecino) solicitando información acerca de la existencia de individuos en la zona.

Para controlar la dinámica del modelo el simulador utiliza mensajes para la comunicación entre *LPs*. Hasta el momento, los mensajes necesarios para llevar a cabo el proceso de selección de vecinos y cálculo de las nuevas posiciones y velocidades son:

- EvRequest
- EvAnswer
- EvMigration

Si bien con estos tres mensajes quedan resueltas las cuestiones del cálculo de las posiciones y velocidades, son necesarios otros mensajes para la sincronización entre los procesadores y el funcionamiento de la máquina de estado que rige la simulación (figura 27).

El mensaje enviado para solicitar información acerca de la existencia de individuos en la zona cercana a la frontera se denomina EvRequest. El LP_{i+1} (bloque receptor de éste mensaje) responderá con un mensaje EvAnswer dando las posiciones de los individuos que se encuentran en la zona. De esta forma, el LP_i (que está calculando la nueva posición y velocidad del *i-ésimo* individuo) tendrá la información completa de todos los vecinos potenciales que se encuentran tanto en su mismo bloque como en el bloque contiguo.

Como resultado del cálculo de la nueva posición puede suceder que ésta sea en el bloque vecino por lo que el individuo se ha desplazado al bloque vecino. Esto implica la migración de un individuo de un bloque a otro. Por lo tanto, este pez es eliminado de las listas del bloque al que pertenecía antes de migrar (LP_i) y es agregado a las listas del bloque que contiene la nueva posición (LP_{i+1}). La forma en la que se implementa esta migración es por medio de un mensaje del tipo EvMigration.

Detalles de la implementación

En el desarrollo del simulador se han considerado a los eventos como mensajes. De esta forma, el núcleo de simulación tiene que seleccionar entre eventos internos y mensajes para generar los cambios en la máquina de estado (figura 27). Dicha selección se realiza de manera tal que se respete la causalidad en todo momento lo cual es impuesto por el método de simulación conservativo utilizado.

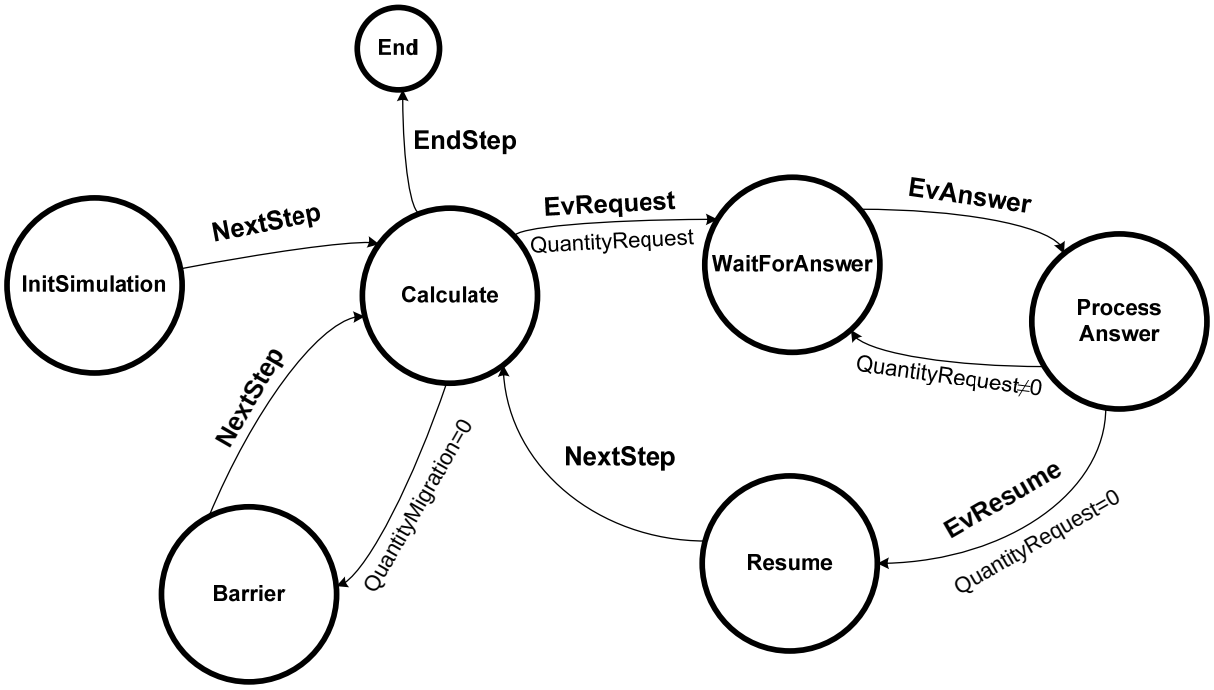


Figura 27: Máquina de estados

El funcionamiento del simulador en una iteración puede ser descrito básicamente por cuatro etapas (figura 28). En la primera se realiza el cálculo de las posiciones y velocidades para los individuos que se encuentran a una distancia mayor que el radio máximo de influencia respecto a la frontera lateral más próxima. Los peces ubicados en esta zona solamente necesitarán saber la posición del resto de los individuos de su bloque. Además del cálculo

mencionado anteriormente, se generan los mensajes de migración y de petición de información a los bloques vecinos para los peces que se encuentran cerca de las fronteras.

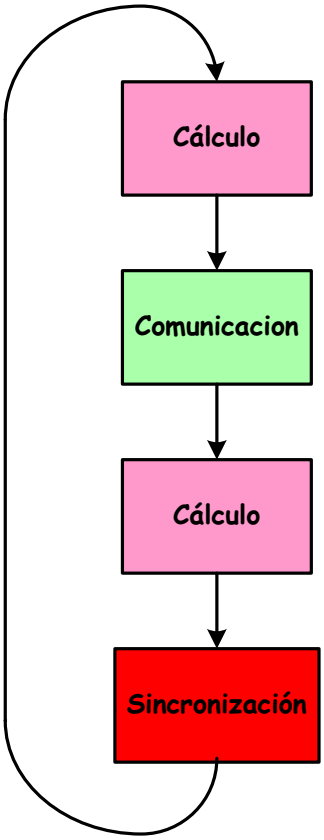


Figura 28: Distintas etapas del simulador

La segunda es una etapa de comunicaciones y de sincronización implícita. Aquí se envían los mensajes generados en la etapa previa y se responden a las peticiones hechas por los *LPs* vecinos.

En resumen, las comunicaciones que se realizan son las siguientes:

- Envío y recepción de EvRequest
- Envío y recepción de EvAnswer
- Envío de EvMigration

El cómputo desarrollado en esta etapa es generado por la respuesta de los mensajes EvRequest y el almacenamietnto de las respuestas EvAnswer. El responder a un mensaje implica el cálculo de la distancia entre el pez indicado por el mensaje y el resto de los peces del bloque consultado.

Aunque en esta etapa no se realiza ninguna sincronización explícita, como podría ser una barrera entre los diferentes *LPs*, se produce una sincronización implícita. Esto es resultado de la condición impuesta a la segunda etapa, a la que no se le permite avanzar con la simulación hasta el momento en el que se hayan enviado y recibido todos los mensajes. La información de cuantos mensajes se deben enviar (en forma de respuestas o preguntas) es una información explícita que viene dada por un mensaje específico.

La tercera etapa es, nuevamente, una etapa de cálculo similar a la primera. La diferencia entre ambas radica en que esta última no genera mensajes de petición de información del tipo EvRequest. En esta etapa, la selección de vecinos potenciales se realiza analizando la posición de todos los peces que se encuentran en el bloque y los individuos provenientes de las respuestas EvAnswer.

Por último, la cuarta etapa es la encargada de recibir/enviar los mensajes de migración y de sincronizar en forma explícita el simulador.

En el próximo apartado se analizará en detalle cada una de las cuatro partes por las que tiene que pasar el simulador en cada iteración.

Simulador, detalles de las etapas

En la figura 29 se observa un diagrama de flujo detallado del simulador sobre el cual se analizarán con más profundidad las distintas etapas por las que pasa el simulador en cada iteración, así como la necesidad de los mensajes de sincronismo para garantizar la causalidad.

El primer paso en la simulación es la inicialización de los distintos procesos lógicos con las posiciones y velocidades de los individuos que pertenecerán a ese bloque. Estos datos son almacenados en cada *LP* en el vector denominado NewFishVector.

Al iniciarse una iteración, el vector NewFishVector es copiado en el vector FishVector y todos los elementos del primer vector son eliminados del mismo. En FishVector se encuentran los individuos con las posiciones y velocidades iniciales para la iteración en curso.

Para cada uno de los elementos de FishVector se analiza su distancia respecto a la frontera con los bloques vecinos. Si la distancia es menor que el radio de influencia mayor, entonces existe la posibilidad de la presencia de vecinos potenciales en el bloque contiguo y se necesitará utilizar un mensaje EvRequest.

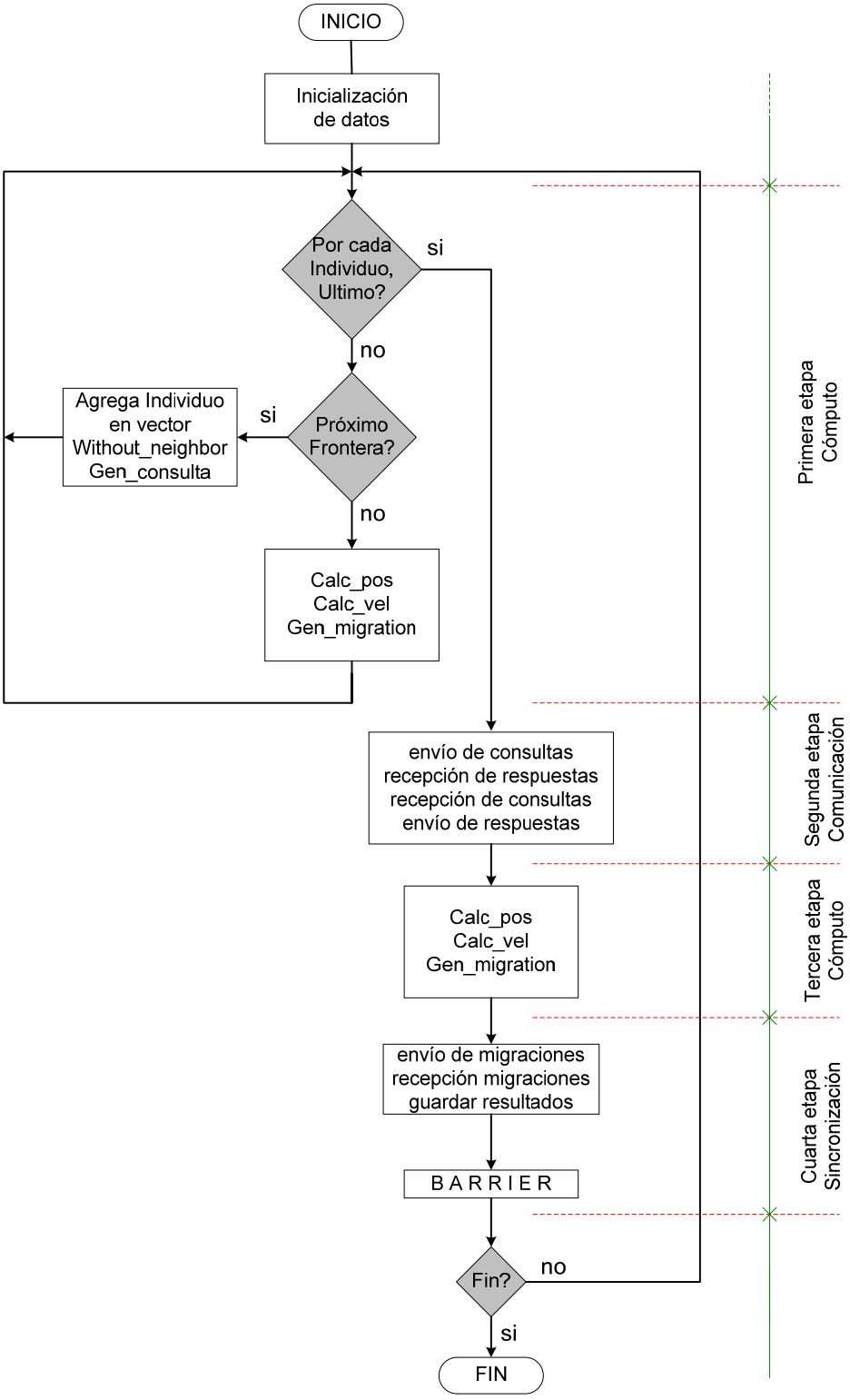


Figura 29: Diagrama de flujo del simulador

Un mensaje EvRequest consta de un identificador que determina el tipo de mensaje, la posición del individuo y el radio máximo:

(EvRequest, Pos, RadMax)

En el caso de generarse el mensaje, éste será enviado al bloque vecino y el individuo al cual se está calculando su nueva posición (a partir de esta momento se hará referencia a él como el *i-ésimo individuo* o *pez* indistintamente) será almacenado en el vector WithoutNeighbour a la espera de recibir la información del bloque vecino acerca de la existencia o no de algún individuo.

Por cada mensaje EvRequest se incrementa el contador mensajes_enviados. Además de este contador existen dos más, cada uno destinado a un bloque vecino, en el cual se lleva la cuenta de la cantidad de mensajes que se enviarán a cada uno de los bloques. Finalizada la generación de todos los mensajes EvRequest se crea y envía un mensaje a cada uno de los bloques vecinos con la cantidad de consultas que recibirá. La información de este mensaje es utilizada para la sincronización de los LPs.

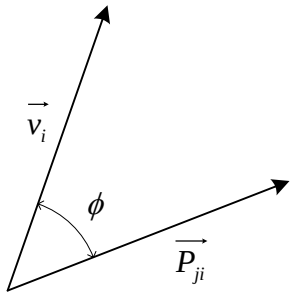
Si no se producen mensajes del tipo EvRequest la selección de los vecinos potenciales se realiza seleccionando entre todos los individuos del bloque al que pertenece, los que se encuentren confinados en el espacio determinado por el radio mayor de influencia y fuera de la zona de visión nula (ver capítulo anterior). Los vecinos seleccionados son almacenados en el vector PotentialNeighbour.

En el seudo código que se muestra a continuación puede verse el proceso de selección de vecinos potenciales descrito anteriormente para el caso en el que la distancia a la frontera del *i-ésimo* pez es mayor que el radio máximo de influencia:

```
mensajes_enviados = 0;
for (por cada individuo F_i de FishVector)
  {
    if (distancia_frontera de F_i < MaxRad)
      {
        generar EvRequest;
        guardar acutal_fish en WithoutNeighbour;
        mensajes_enviados++;
      }else
        for (cada elemento F_j de FishVector ≠ F_i)
          {
            if (distancia (F_i, F_j)<MaxRad && F_i ⊄ DA)
              {
                agregar  F_j en PotentialNeighbour;
              }
          }
  }
enviar EvRequests
```

Una vez seleccionados los vecinos potenciales, se deben escoger los cuatro que se encuentren más cerca de la visión directa del *i-ésimo* pez. Este procedimiento se realiza mediante el siguiente algoritmo:

- Para cada elemento del vector PotentialNeighbour se calcula el producto escalar entre los vectores unitarios $\vec{v_i}$ y el formado por $\vec{P_j} - \vec{P_i} = \vec{P_{ji}}$. El valor obtenido estará contenido en un rango de valores entre -1 y 1. Los valores más próximos a 1 corresponden a los individuos que se encuentran más cerca de la visión directa del *i-ésimo* pez y el resultado se vincula al *j-ésimo* pez (figura 30).



$$\vec{v_i} \cdot \vec{P_{ji}} = \left|\vec{v_i}\right| \left|\vec{P_{ji}}\right| \cos(\phi)$$

como $\vec{v_i}$ *y* $\vec{P_{ji}}$ *son unitarios*

$$\vec{v_i} \cdot \vec{P_{ji}} = \cos(\phi)$$

Figura 30: Uso del producto escalar para la selección de los vecinos

- Una vez obtenidos los resultados de todos los productos vectoriales, se ordenan en forma creciente por la distancia en relación a la visión directa del *i-ésimo* individuo.

- A partir de este punto se escogen los cuatro primeros elementos del vector. Dado que estos están ordenados serán los mejor posicionados y con los que interactuará el *i-ésimo* pez. Cabe destacar que esta forma de selección es un caso general ya que podría suceder que más de un vecino tenga el mismo producto escalar. Esta situación puede visualizarse en dos dimensiones como muestra la figura 31.

Los individuos *j*, *k* y *l* tienen un mismo ángulo α de desviación respecto a la visión directa del *i-ésimo* individuo. En este caso, el ordenamiento no sólo tiene en cuenta el ángulo sino también la distancia entre el *i-ésimo* individuo y los demás vecinos, seleccionando al que tiene menor distancia. En el ejemplo de la figura 31 entre el individuo *k* y *l*, se seleccionará el *k*. Esto se debe a que el individuo *l* no será visto por el *i-ésimo* pez.

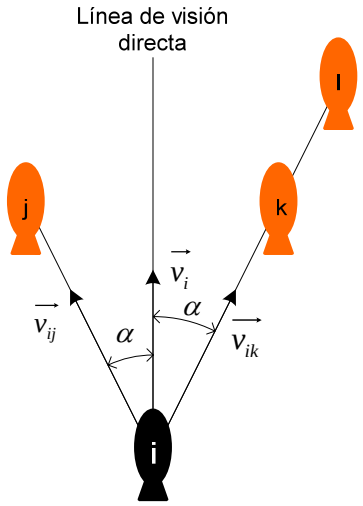


Figura 31: Caso particular de varios vecinos con el mismo ángulo de respecto a la visión directa del i-ésimo pez

Una vez seleccionados los cuatro vecinos más cercanos a la línea de visión directa, se calcula la nueva velocidad y posición y se almacena el individuo en el vector NewFishVector. Si la nueva posición no se encuentra en el mismo bloque se genera un mensaje EvMigration con el siguiente formato:

(EvMigration, Posición, Velocidad)

Luego que se han realizado los pasos mencionados, se entra en un el estado de envío/espera de repuestas y consultas (WaitForAnswer en la máquina de estados). Al inicio de este estado se reciben mensajes de los bloques vecinos con la información de la cantidad de consultas que se realizarán. Una vez realizada la recepción de estos mensajes, se reciben el resto de los mensajes y se los procesan.

Al finalizar esta etapa, se inicia el procedimiento de selección de vecinos para el cálculo de las nuevas posiciones y velocidades para los individuos que se encuentran en el vector WithoutNeighbour.

En el caso de producirse una migración se generan los mensajes EvMigration correspondientes y estos mensajes, junto con los generados en la primera etapa de cálculo de posiciones y velocidades, son enviados a los procesos lógicos correspondientes.

Los elementos que han migrado son eliminados del vector FishVector puesto que en la próxima iteración ellos no pertenecerán a ese proceso lógico. Cuando se recibe un mensaje del tipo EvMigration, el individuo que ha migrado es anexado al vector NewFishVector.

Al finalizar la recepción de todos los mensajes de migración se verifica la condición de finalización de la simulación. Si no se ha llegado al final se copia el vector NewFishVector en FishVector, se eliminan todo los elementos del primero y se reanuda la simulación.

Si se ha llegado al final de la simulación se consume el evento EndStep y la simulación es finalizada.

Sincronización para el método conservativo

La simulación distribuida utilizando el método conservativo implica que la causalidad tiene que ser respetada en todo momento. Para cumplir este requerimiento es necesaria la implementación de un método de sincronización entre los procesos lógicos que forman la simulación para que todos tengan el mismo tiempo de simulación en el momento del envío de los mensajes. De no suceder lo comentado anteriormente se pueden dar situaciones en las que un proceso lógico avance en el tiempo más que los otros y que reciba un mensaje con tiempo en el pasado es decir, menor a su tiempo de simulación actual lo que implica incoherencia de la simulación.

En el desarrollo del simulador se han implementado mensajes que tienen como única función la sincronización. Estos mensajes son:

- EvQuantityMessages
- EvQuantityMigrations

Antes del envío de los mensajes EvRequest, se envía un mensaje EvQuantityMessages en el que se indica la cantidad de mensajes de consulta que serán enviados. De esta forma tanto el proceso lógico receptor y el generador de las consultas tienen conocimiento de cuántos mensajes tienen que responder y esperar respectivamente. Una vez que cada proceso lógico responde todos los mensajes y recibe todas las respuestas, puede continuar con la etapa 3 de cálculo de nuevas posiciones y velocidades (figura 29).

Para el caso de las migraciones, la situación es análoga con la diferencia de que, a continuación de la recepción de todos los nuevos individuos de los bloques vecinos, se pasa a un estado de sincronización forzada por una barrera (barrier). De no existir esta última sincronización y el mensaje EvQuantityMigrations, los procesos que terminan antes la recepción de los nuevos individuos comenzarán con la nueva iteración y las migraciones que

lleguen a destiempo se agregarán al vector NewFishVector correspondiente a una iteración posterior.

Finalizado el análisis del funcionamiento del simulador, en el próximo apartado se realizará una descripción matemática de las diversas etapas del simulador.

4.4. Modelo analítico y de prestaciones del simulador

Como se ha visto anteriormente, el funcionamiento del simulador se divide en cuatro etapas. Cada una de éstas consume una determinada cantidad de tiempo que se encuentra estrechamente ligada a las características de la simulación realizada y a la implementación del simulador.

Para el análisis de las prestaciones y el modelado analítico del simulador se consideró que el tamaño de los procesos lógicos del simulador son iguales. Lo mismo se ha hecho con la cantidad de individuos en cada bloque. La distribución de los peces se hizo en forma aleatoria en todo el volumen de los *LPs*. De esta forma se tiene el sistema balanceado, cada procesador que simula un proceso lógico, tiene la misma carga computacional que los demás.

El tiempo de simulación por iteración se definirá de la siguiente forma:

"T_i = *tiempo necesario para realizar el cálculo de todas las posiciones y velocidades de todos los individuos del sistema*".

Este tiempo es la suma de los tiempos utilizados por cada una de las cuatros etapas (figura 29):

$$T_i = T_1 + T_2 + T_3 + T_4 \tag{40}$$

Antes de continuar con el análisis del tiempo empleado para realizar una iteración, es conveniente hacer ciertas definiciones de términos que se utilizarán en el resto del capítulo. En la figura 32 se puede observar que el ancho de un LP es de *a* [BL]. Los individuos que se encuentran en los paralelepípedos 1 son los que necesitan realizar consultas a los bloques vecinos para poder determinar sus nuevas posiciones y velocidades. A esta cantidad de individuos se la denominará *C* y *N* a la cantidad total de peces en el LP. En la zona 2 se encuentran los peces que no necesitan realizar consultas a los bloques vecinos puesto que su

distancia a las caras laterales es mayor que R_3. La cantidad de peces en esta zona es proporcional a N-C.

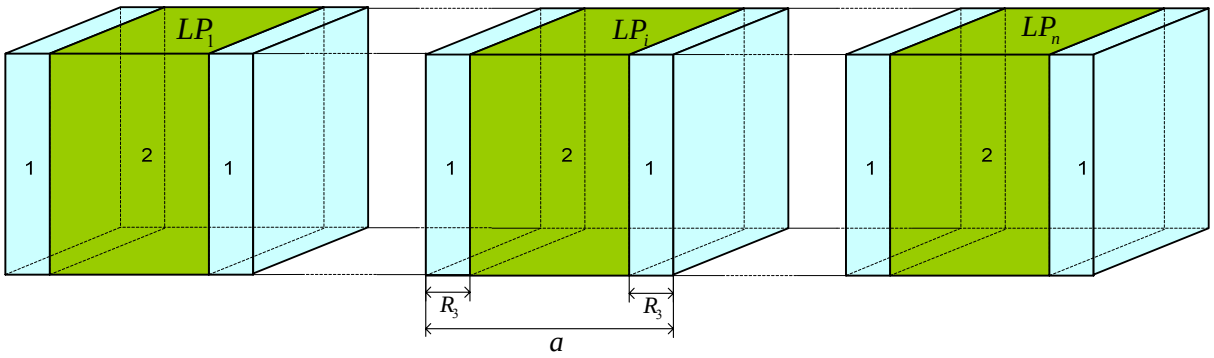


Figura 32: Distribución del espacio y regiones de envío de preguntas

Como se ha dicho anteriormente, la primera etapa se basa en el cálculo. El tiempo T_1 es usado en realizar el cómputo de la distancia de cada elemento con el resto de los individuos y calcular la nueva posición y velocidad. Esto se hace para los peces que se encuentran en la zona 2. La complejidad de esta etapa es:

$$O\big((N-C)(N-1)\big) \tag{41}$$

Como se trabaja con grandes cantidades de individuos por bloque, se puede realizar la siguiente aproximación:

$$O\big((N-C)(N-1)\big) \approx O\big((N-C)N\big) \tag{42}$$

El tiempo T_1 estará determinado por:

$$T_1 = K_1(N-C)N \tag{43}$$

Al encontrarse todos los peces distribuidos en forma aleatoria en todo el volumen del LP, C se puede considerar proporcional a $\frac{2R_3}{a}N$. Por lo tanto C será igual a:

$$C = K_{1_1}\frac{2R_3}{a}N \tag{44}$$

Reemplazando (44) en (43) se tiene:

$$T_1 = K_1\left(N - K_{1_1} \frac{2R_3}{a} N \right)N$$

$$\boxed{T_1 = K_1\left(1 - K_{1_2} \frac{R_3}{a} \right)N^2} \quad (45)$$

$$\text{con } K_{1_2} = 2K_{1_1}$$

Los valores de las dos incógnitas, K_1 y K_{1_2}, es posible obtenerlos conociendo datos que son tanto del modelo simulado como de los parámetros de la simulación ejecutada. Dichos valores son el tiempo utilizado en la primera etapa (T_1), el ancho del proceso lógico (a), el radio mayor de influencia (R_3) y la cantidad de individuos en el proceso lógico (N). Este último valor también puede estar vinculado con la cantidad de procesadores, si se expresa la cantidad de individuos (n) como el total que se encuentra en todo el espacio de simulación. En este caso, la expresión de T_1 puede reescribirse como:

$$\boxed{T_1 = K_1\left(1 - K_{1_2} \frac{R_3}{a} \right)\left(\frac{n}{P} \right)^2} \quad (46)$$

donde P es la cantidad de procesadores.

El tiempo T_2 es el más complejo de analizar puesto que en él intervienen diversos factores. En la segunda etapa se realiza el envío y recepción de mensajes necesarios para el cálculo de las nuevas posiciones y velocidades.

T_2 puede ser expresado como la suma de los siguientes términos:

$$\boxed{T_2 = T_{2R} + T_{2C}} \quad (47)$$

donde:

- T_{2R} = tiempo para responder a las preguntas Evrequest.
- T_{2C} = tiempo utilizado en realizar las comunicaciones y para que se cumplan las condiciones necesarias para continuar con la simulación.

En T_{2R}, por cada una de las consultas provenientes de los *C* individuos, se debe calcular la distancia entre el individuo que consulta y los *N* que se encuentran en el bloque. Además se genera el mensaje de respuesta con los individuos encontrados. El tiempo T_{2R} puede ser expresado de la siguiente forma:

$$T_{2R} = K_{2R} CN$$

$$T_{2R} = K_{2R}\left(2\frac{R_3}{a}N\right)N$$

$$\boxed{T_{2R} = K_{2R'}\frac{R_3}{a}N^2} \qquad (48)$$

$$\text{con } K_{2R'} = 2K_{2R}$$

T_{2C} es un tiempo que depende de diversos factores, algunos de los cuales no responden a una complejidad algorítmica determinada. Por tal motivo se ha hecho uso de datos empíricos para la construcción de la ecuación que representa el comportamiento del simulador en esta etapa.

En el tiempo T_{2C} se envían y reciben los mensajes de consulta/respuesta. Los factores que intervienen son el tiempo de envío de los *C* mensajes y el tiempo de cómputo para la respuesta de éstos. Mediante la interpolación de los tiempos de T_{2C} obtenidos en las experimentaciones se encontraron las constantes del polinomio de tercer orden que lo representará. Como variable se ha utilizado la cantidad de mensajes enviados $C = 2R_3N/a$. En (49) se muestra el polinomio expresado en formas genéricas y en (50) con las constantes obtenidas.

$$\boxed{\begin{aligned} T_{2C} &= K_{2C_1}C^3 + K_{2C_2}C^2 + K_{2C_3}C + K_{2C_4} \\ T_{2C} &= K_{2C_1}\left(\frac{2R_3}{a}N\right)^3 + K_{2C_2}\left(\frac{2R_3}{a}N\right)^2 + K_{2C_3}\left(\frac{2R_3}{a}N\right) + K_{2C_4} \end{aligned}} \qquad (49)$$

$$\boxed{T_{2C} = -1{,}56\times10^{-10}\left(\frac{2R_3}{a}N\right)^3 + 5{,}73\times10^{-6}\left(\frac{2R_3}{a}N\right)^2 - 7{,}5\times10^{-3}\left(\frac{2R_3}{a}N\right) + 6{,}106} \qquad (50)$$

Por lo tanto, T_2 puede reescribirse:

$$\boxed{T_2 = K_{2R'}\frac{R_3}{a}N^2 + K_{2C_1}\left(\frac{2R_3}{a}N\right)^3 + K_{2C_2}\left(\frac{2R_3}{a}N\right)^2 + K_{2C_3}\left(\frac{2R_3}{a}N\right) + K_{2C_4}} \quad (51)$$

y teniendo en cuenta que $N = {}^{n}\!/_{P}$, donde *n* es la cantidad total de individuos simulados y *P* la cantidad de procesadores (la misma que de LPs), T_2 será:

$$\boxed{T_2 = K_{2R'}\frac{R_3}{a}\left(\frac{n}{P}\right)^2 + K_{2C_1}\left(\frac{2R_3}{a}\frac{n}{P}\right)^3 + K_{2C_2}\left(\frac{2R_3}{a}\frac{n}{P}\right)^2 + K_{2C_3}\left(\frac{2R_3}{a}\frac{n}{P}\right) + K_{2C_4}} \quad (52)$$

El tiempo T_3 de la tercera etapa es el tiempo utilizado para el cálculo de las velocidades y posiciones de los *C* individuos. Para tal fin, se tienen que comparar la distancia entre cada uno de los *C* individuos con los N que pertenecen al bloque más los *R* que se encuentran cerca de las fronteras ($R \propto \frac{R_3}{a}N$). Por lo tanto el tiempo T_3 puede escribirse de la siguiente forma:

$$T_3 = K_3 C(N + R)$$

$$T_3 = K_3 2\frac{R_3}{a}N\left(N + K_{3a}\frac{R_3}{a}N\right)$$

$$\boxed{T_3 = K_{3b}\frac{R_3}{a}N^2 + K_{3c}\left(\frac{R_3}{a}\right)^2 N^2} \quad (53)$$

$$\text{con } K_{3b} = 2K_3 \text{ y } K_{3c} = 2K_{3a}K_3$$

y como $N = {}^{n}\!/_{P}$, puede expresarse también como:

$$\boxed{T_3 = K_{3b}\frac{R_3 n^2}{aP^2} + K_{3c}\left(\frac{R_3 n}{aP}\right)^2} \quad (54)$$

En la última etapa se realiza la sincronización necesaria para mantener la coherencia entre los tiempos de simulación de los diferentes bloques y la eliminación de situaciones de existencia de más de un pez en un mismo punto del espacio.

La sincronización se realiza por medio de barreras y el procesamiento/modificación de posiciones iguales de individuos por medio de un algoritmo de complejidad $O(N^2)$, siendo N la cantidad de peces en el bloque.

Por lo tanto, el tiempo utilizado en esta sección del simulador puede ser modelado de la siguiente forma:

$$\boxed{T_4 = K_4 N^2} \tag{55}$$

$$\boxed{T_4 = K_4 \left(\frac{n}{P}\right)^2} \tag{56}$$

Una vez obtenidos todos los términos de la ecuación que representa el tiempo de simulación por iteración, queda por obtener la fórmula final. Si se considera que se utilizan P procesadores y que el ancho de cada proceso lógico es de a, el ancho total del espacio de simulación queda definido por $L = aP$. El tiempo total por iteración escribiendo a en función del número de procesadores será:

$$T_i = T_1 + T_2 + T_3 + T_4$$

$$\boxed{T_i = \frac{n^2 R_3}{PL} K_a + \frac{n^2}{P^2} K_b + \frac{n^2 {R_3}^2}{L^2} K_c + \frac{n^3 {R_3}^3}{L^3} K_d + \frac{n R_3}{L} K_e + K_f} \tag{57}$$

donde:

$$K_a = K_{2R'} + K_{3b} - K_1 K_{1_2},\ K_b = K_1 + K_4,\ K_c = 4K_{2c_2} + K_{3c},$$
$$K_d = 8K_{2c_1},\ K_e = 2K_{2c_3},\ K_f = K_{2c_4}$$

Con esta fórmula, se puede obtener el tiempo de simulación por iteración en base a:

- Parámetro del modelo biológico: R_3.
- Parámetros de la simulación: n, L y P.

Se puede concluir que conociendo los recursos disponibles (P), la cantidad de individuos y el tamaño de espacio de simulación es posible realizar una estimación del tiempo que se requiere para ejecutar la simulación con las características anteriormente mencionadas.

En muchas ocasiones es de interés saber que recursos serán necesarios para realizar una simulación de ciertas características y con una determinada restricción en el tiempo por iteración.

Otra situación que también presenta interés es poder estimar la cantidad de individuos que se pueden simular en un tiempo dado y con unos recursos conocidos.

Las fórmulas que permiten conocer los recursos necesarios y la cantidad de peces a simular en función de las características de la simulación serán desarrolladas a continuación.

Los recursos necesarios, es decir, la cantidad de procesadores requeridos puede expresarse de la siguiente forma:

$$\boxed{P^2 A_P + P B_P + C_P = 0} \tag{58}$$

donde:

$$A_P = L\left(T_i - K_c\left(\frac{R_3 n}{L}\right)^2 - K_d\left(\frac{R_3 n}{L}\right)^3 - K_e \frac{R_3 n}{L} - K_f \right),\ B_p = R_3 n^2 K_a,\ C_p = -n^2 L K_b,$$

Resolviendo el polinomio se encuentra la cantidad de procesadores para las condiciones impuestas de número de individuos y dimensiones del espacio de simulación. De los resultados obtenidos, solamente uno será el correcto, el otro carece de sentido físico.

Por último, la cantidad de individuos que pueden simularse en un tiempo dado y con determinados recursos se calcula mediante el siguiente polinomio:

$$\boxed{n^3 A_n + n^2 B_n + n C_n + D_n = 0} \tag{59}$$

donde:

$$A_n = \left(\frac{R}{L}\right)^3 K_d,\ B_n = \frac{R_3}{PL} K_a + \frac{1}{P^2} K_b + \left(\frac{R}{L}\right)^2 P K_c,\ C_n = \frac{R_3}{L} K_e,\ D_n = K_f - T_i$$

Nuevamente, hallando las raíces del polinomio de tercer grado se obtiene la cantidad de individuos. Puesto que existirán tres soluciones, solamente una será la que corresponda al valor buscado, las demás soluciones carecen de sentido (números negativos o complejos).

Como se puede observar en las ecuaciones desarrolladas anteriormente, cuando se hace referencia a las dimensiones del espacio de simulación sólo se tiene en cuenta el ancho. La profundidad y la altura deben ser tales que la densidad de individuos sea la densidad que existiría en estado estacionario, es decir cuando los peces se mueven formando un banco [37] [36] [19] [56].

Lenguaje de programación y librería de comunicación utilizados

Puesto que los lenguajes orientados a objetos permite una gran flexibilidad a la hora de codificar los simuladores, se ha optado como lenguaje de programación C++ y las librerías estándar STL para el manejo de las listas y las colas de eventos, así como también para los buffers de entrada/salida.

Las comunicaciones en sistemas distribuidos pueden realizarse a distintos niveles, desde el más bajo, con librerías de comunicación estándar [69] (BSDSockets), o a niveles superiores, con librerías de comunicación dedicadas como MPI [55] o PVM [30].

El utilizar comunicaciones a bajo nivel tiene como beneficio una mayor eficiencia mientras que la complejidad en su codificación y depuración son mayores. Las comunicaciones realizadas a niveles superiores implican una ventaja en el momento de su codificación y en el análisis lógico de la estructura de comunicaciones. Como punto negativo, en cambio, se tiene una disminución de las prestaciones.

Desarrollar una aplicación donde es necesaria una abstracción del sistema de comunicaciones, como es el caso de la simulación distribuida, el utilizar comunicaciones de bajo nivel implica una codificación extremadamente compleja y la pérdida de la visión global de la lógica de simulación.

Con el fin de encapsular las comunicaciones y teniendo en cuenta la pérdida de eficiencia, se ha optado por la segunda opción basada en comunicaciones por paso de mensajes.

Un programa de paso de mensajes consiste de múltiples procesos, cada uno compuesto de código secuencial al que se le han incluido funciones o rutinas para enviar y / o recibir mensajes. Los procesos en los programas de paso de mensaje se ejecutan en forma asincrónica y cooperan en la resolución de un problema. Las interacciones deben estar en forma explícita en los programas al igual que el paralelismo.

Para la implementación de las comunicaciones se utilizó la librería de comunicaciones MPI. Esta decisión ha sido tomada puesto que experimentaciones realizadas anteriormente demostraron que la utilización de PVM generaba incrementos de tiempo y problemas en la escalabilidad cuando la cantidad de individuos y procesadores crecía [72]

4.5. Alternativa al modelo de simulación clásico

El simulador fue desarrollado utilizando una distribución a nivel de componentes debido a las características espaciales que presenta el modelo (Capítulo 2). El espacio que ocupan los individuos se dividió en procesos lógicos. Cada uno de éstos representa una división del *espacio simulado o de simulación* y es asignado a un procesador.

Tanto los peces como otros seres biológicos o individuos que pueden ser modelados y simulados en forma análoga a la realizada en la presente tesis, manifiestan un desplazamiento espacial durante el transcurso de la simulación. Puesto que el *espacio simulado* es finito, el movimiento del banco de peces puede ir más allá de las fronteras preestablecidas. En la figura 33 se puede ver un banco de peces en el inicio de la simulación y cual sería la posición en la *i-ésima* iteración. Como se observa, el banco de peces se tendría que haber desplazado fuera de los límites del espacio de simulación. Esta situación acarrea el problema de la forma en la que se deben tratar las nuevas posiciones que se hallan fuera del espacio de simulación predeterminado.

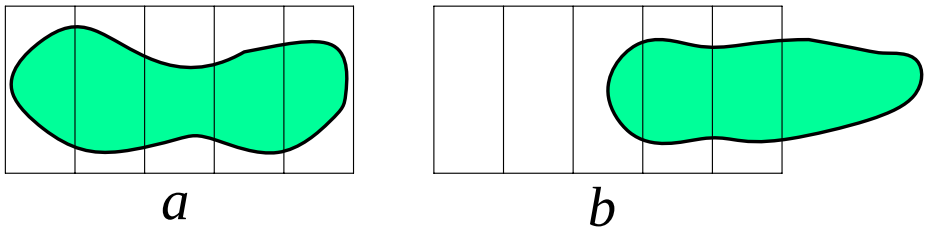


Figura 33: Posición de un banco de peces en a) inicio de la simulación y en b) en la iteración *i-ésima* iteración

En la implementación realizada, los peces que se encuentran en esta situación vuelven a entrar por la cara opuesta a la que salieron (figura 34). De esta forma, la densidad de individuos en el espacio de simulación se mantiene constante a lo largo de toda la simulación. Aunque esto no representa completamente la realidad puesto que los individuos que son reincorporados no forman más parte del grupo original, el modelo es igualmente útil para explicar y justificar las

propuestas que se harán respecto a la forma en la cual se puede tratar la problemática de la distribución del modelo y de la carga en los procesadores.

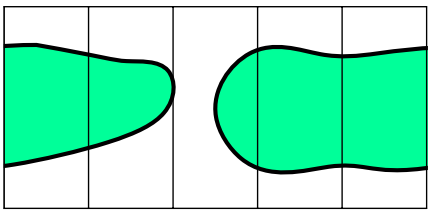


Figura 34: Reincorporación de individuos cuando se hallan fuera del mundo simulado

En el próximo capítulo dedicado a la experimentación se mostrarán las pruebas realizadas para validar el modelo. Para tal fin, no se ha hecho una distribución de individuos equiprobable sobre todo el espacio, sino que se han replicado las pruebas en grupos pequeños tal como lo describen los autores del modelo biológico [37] [36]. De esta forma se permiten grandes desplazamientos en el espacio de simulación para lo cual se colocan los peces de manera tal que se encuentren lejos de las fronteras.

Para la solución del problema de los límites estáticos impuestos en la división del espacio simulado y los efectos de la reinserción de los peces, se desarrollarán distintas alternativas que concluirán en la propuesta que se realizará para la distribución, balanceo y mayor ganancia de tiempo en simulaciones de modelos orientados al individuo.

Como la mayoría de las especies de peces se mueven en bancos, una alternativa para el problema de las fronteras sería hacer un seguimiento del grupo para identificar sus límites y, de esta forma, acompañar el movimiento con los procesadores en los cuales ha sido dividido el banco de peces.

En la figura 35 se muestra un posible caso en el que el simulador ha seguido al grupo en los primeros cuatro tiempos de simulación. Como se puede observar, aunque el banco de peces se ha desplazado, no se ha reincorporado ningún pez por encontrarse fuera del espacio de simulación. En este caso, este espacio se modifica en forma dinámica con el paso del tiempo y de la dinámica del grupo.

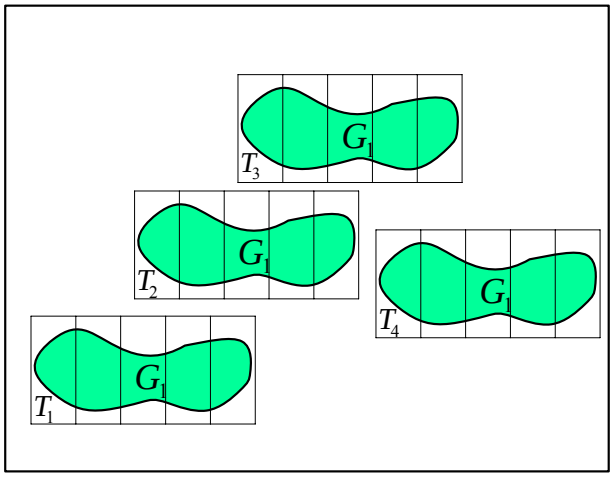


Figura 35: Seguimiento de un banco de peces

Un espacio de simulación con más de un banco de peces es un escenario que presenta interés para el análisis de la interacción entre ellos además de ser una situación más general que se presenta en la naturaleza. Si por ejemplo se tienen cuatro grupos de individuos y cinco procesadores, continuado con la filosofía de la distribución del sistema físico (la misma que la utilizada en el simulador desarrollado), se tendría una situación como la que se puede apreciar en la figura 36, en la que existe un desbalanceo de carga entre los diferentes procesadores. Esta diferencia en la carga puede incrementarse aún más si, por ejemplo, los grupos, de acuerdo a la dinámica impuesta por el comportamiento de sus individuos, se desplazan y quedan posicionados como se muestra en la figura 37. El desbalanceo es muy elevado, prácticamente todos los individuos se encuentran en los procesadores P_3 y P_4. El procesador P_5 sólo tiene una porción muy reducida de G_3 y G_4. Si se utilizara el simulador implementado para una situación como ésta, el tiempo por iteración estaría limitado por el tiempo requerido por los procesadores P_3 y P_4. El resto de los procesadores se encontrarán ociosos hasta el momento en el que, por la dinámica del sistema, todos los procesadores vuelvan a quedar balanceados. Esta situación debe evitarse.

Además del problema de la carga, los individuos que lleguen a un límite del espacio de simulación se reincorporarán por la cara opuesta a la que salieron, alejándose de la realidad como se comento en el inicio de este apartado.

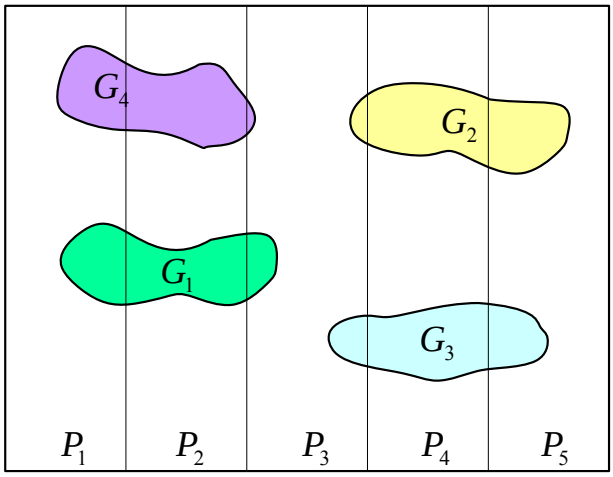


Figura 36: Distribución del espacio con varios bancos de peces

Del problema de la forma en la que se puede realizar una distribución adecuada de la carga en el sistema, y que no sea solamente en el inicio de la simulación y en momentos particulares en los cuales la ubicación transitoria de los individuos hace que la carga esté balanceada, se han desarrollado las ideas que se describen a continuación, y que concluyen con una propuesta alternativa para la distribución de carga.

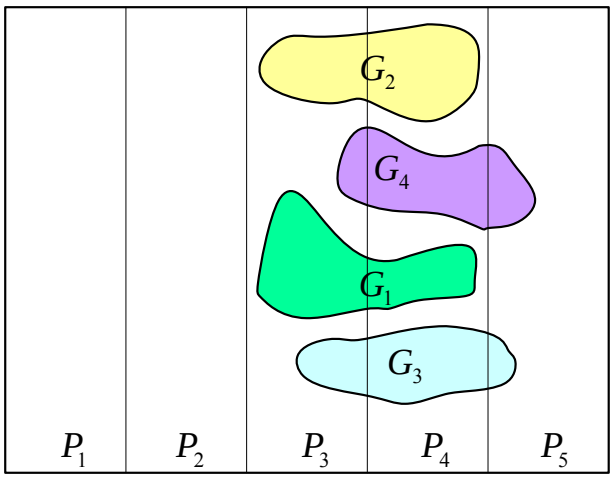


Figura 37: Desbalanceo de carga

Para desarrollar la propuesta se utilizará un escenario genérico de simulación en el que existen diversos bancos de peces distribuidos en él. En la figura 38 puede observarse lo dicho anteriormente, donde varios bancos de peces conviven en un mismo *espacio de simulación*.

Si la simulación de este mundo se realizara con un simulador secuencial, todos los bancos serían calculados por un solo procesador. El tiempo para realizar dicha tarea sería elevado si las cantidades de individuos que se tratan son del orden de los miles o cientos de miles.

Como se mencionó anteriormente, la complejidad del algoritmo básico de simulación es de $O(N(N-1))$. A forma de aproximación se puede considerar que el tiempo que tardará este sistema en realizar la simulación es proporcional a:

$$T_s = KN(N-1) \tag{60}$$

donde N es la cantidad de individuos totales.

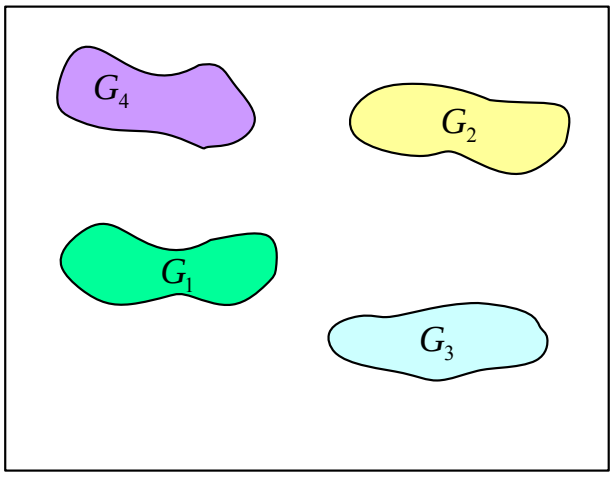


Figura 38: Varios bancos de peces en un mismo mundo de simulación

Generalizando la implementación distribuida mostrada en la figura 35, se podría identificar a cada grupo G_i y dividir el espacio que ocupan en tantos procesadores como se disponga. De esta forma, cada procesador computará una porción de cada banco de peces, como puede observarse en el ejemplo de la figura 39, donde cada grupo es dividido en cinco procesos lógicos. Por lo tanto a cada procesador p le corresponde el cómputo del LP_p de cada grupo.

Una simulación de este tipo requeriría un constante seguimiento de cada uno de los grupos de individuos para poder hacer la división del mismo y actualizar las coordenadas de cada subdivisión (proceso lógico).

Cada procesador, al realizar el cómputo de los individuos que contiene, deberá consultar tanto a los peces que se encuentran en el mismo grupo que el pez en análisis de nueva posición como a los peces que pertenecen a diferentes grupos y que conviven en el mismo procesador. Además, se generarán mensajes a otros procesadores, como se ha descrito en la primera parte de este capítulo.

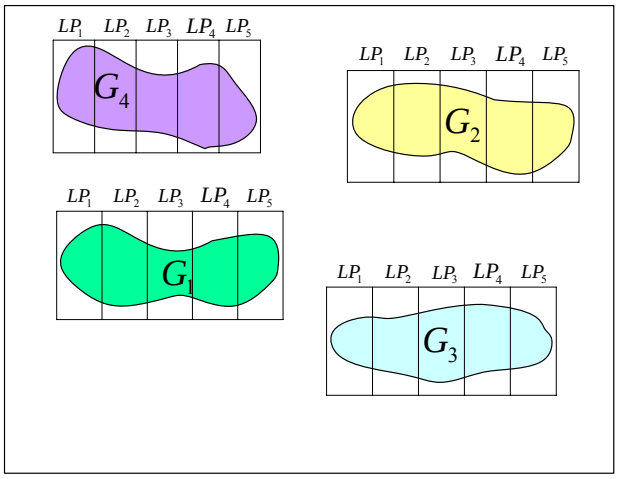


Figura 39: Distribución de los bancos de peces en los procesadores

La principal desventaja que tendría este método para distribuir el modelo sería la necesidad de tener constantemente conocimiento de los límites de cada banco de peces. Esto es debido a que los grupos se van trasladando durante el transcurso de la simulación y cambian su forma. Como ventaja, este seguimiento permitiría un mejor balanceo de carga ya que los procesadores siempre tendrán individuos dentro del subespacio del mundo de simulación asignado. Otra cuestión que se evitaría es la salida e incorporación de los individuos. Cuando llegan a un extremo del espacio asignado al procesador este espacio varía dinámicamente de acuerdo a la forma que toma el grupo.

Una alternativa a la distribución descrita anteriormente, y que evita la carga computacional acarreada por el conocimiento constante de las formas de los grupos, se basa en el mapeado de todo el espacio de simulación en un solo *módulo de cómputo*.

En la figura 40 se puede ver el mapeado del espacio de simulación que consiste en dividir el espacio total en módulos y procesar a todos ellos en un único *modulo de cómputo*. Puesto que este módulo contiene a todo el espacio, todos los bancos de peces se hallan en él.

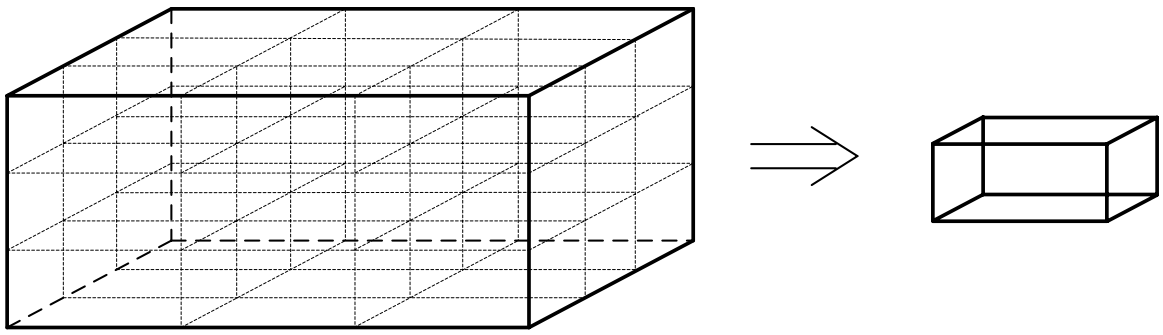


Figura 40: División del espacio de simulación en módulos

La figura 41 es un ejemplo de cómo quedarían mapeados los bancos de las figuras anteriores pero haciendo una proyección en el plano para simplificar la visualización. Como puede observarse, el *módulo de cómputo* es dividido y cada subdivisión de éste es simulada por un procesador. Por lo tanto, cada procesador simulará un proceso lógico de cada uno de los módulos en los que se dividió el espacio de simulación.

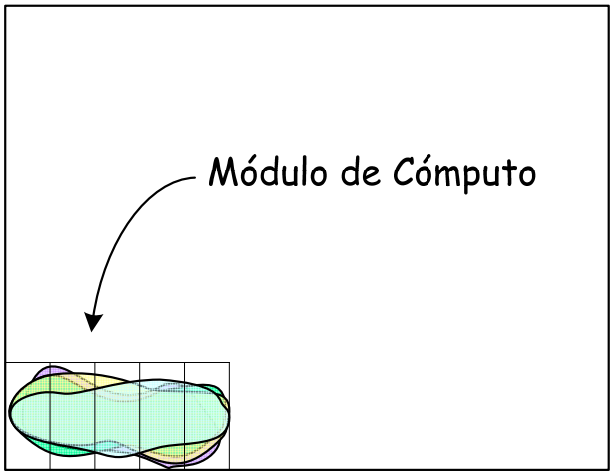


Figura 41: Módulo de Cómputo de cinco procesadores

Las dimensiones del módulo de cómputo son fijas pero no representan solamente a una parte del espacio de simulación sino a todos los módulos que están contenidos en todo el espacio de simulación.

Con esta reducción del sistema completo a un solo espacio se evita la necesidad de contar con la información de las fronteras de cada grupo para poder reasignar las dimensiones de cada proceso lógico. En este caso, cuando un pez cruza una frontera del módulo de cómputo y por ende la del módulo al que pertenecía, es reincorporado por la cara opuesta a la que salió pero con una coordenada que se corresponde al desplazamiento al módulo vecino en el que se hallaba.

La figura 42 es un ejemplo de la migración y la reinserción de un pez que se halla en la frontera del módulo. En la iteración *k* el individuo aún no ha migrado pero en la iteración siguiente (*k+1*) sí lo hace. Se puede ver como el pez continúa su trayectoria establecida en la parte izquierda de la figura 42 la cual se corresponde con una representación bidimensional del *espacio de simulación* (esta forma de representación es sólo para simplificar la comprensión visual de la figura). Como se comentó anteriormente, al estar todo el espacio mapeado en un mismo módulo de cómputo, el pez es reincorporado por la cara opuesta por la

que salió y será simulado por el procesador P_1 que contiene al proceso lógico LP_1 del módulo $i+1$.

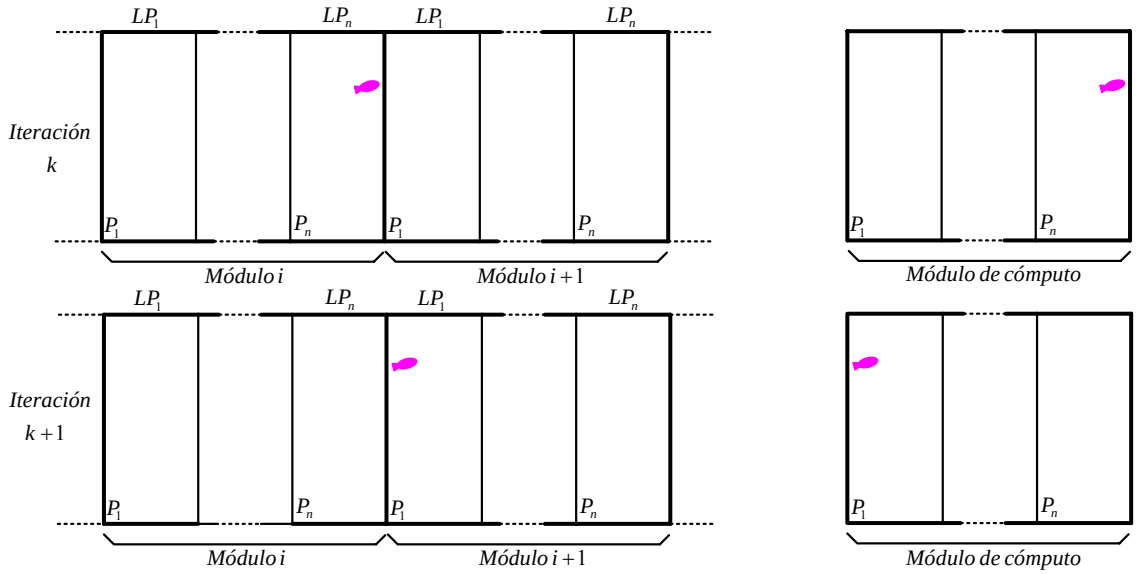


Figura 42: Ejemplo de migración utilizando el *módulo de cómputo*

Esta nueva forma de tratar el espacio de simulación requiere que las coordenadas de los individuos sean reformuladas. A diferencia de las coordenadas utilizadas en los casos mostrados anteriormente, ahora la ubicación de un pez en el espacio de simulación estará dada por una doble coordenada. Los individuos serán reconocidos por coordenadas macroespaciales y locales.

Las coordenadas locales indicarán el punto en el espacio del *módulo de cómputo* donde se halla y las coordenadas macroespaciales harán lo mismo pero para indicar en cuál de los módulos mapeados en el *módulo de cómputo* se halla. La combinación de ambas coordenadas facilitará la detección de que peces se encuentran cerca de otro par interactuar.

Cada individuo además de poseer la doble coordenada podría tener una identificación que determinará a qué especie pertenece. De esta forma no sería necesario que en el momento de determinar la nueva posición y velocidad de cada individuo se tuviera que calcular la distancia entre él y todos los individuos del bloque. Puesto que en el *módulo de cómputo* se encuentran todos los individuos del espacio de simulación, los datos de la especie a la que pertenecen y las macrocoordenadas reducirán el tiempo de cálculo debido a la disminución de la complejidad a simular por cada procesador como se mostrará en breve.

Por último se harán unas aproximaciones para mostrar las diferencias de tiempos que se podrían obtener mapeado en el *módulo de cómputo* el espacio de simulación total y aplicando las dos tipos de coordenadas o no.

El *módulo de cómputo* se encuentra dividido en *P* submódulos, es decir, en la misma cantidad de procesadores con los que se esté trabajando. Si se considera una distribución de peces equiprobable, la cantidad de individuos en cada procesador es:

$$\frac{\sum_{I} g_i}{P} \tag{61}$$

donde:

P = cantidad de procesadores, g_i = número de individuos del grupo G_i, I = número de grupos

Como primera aproximación del tiempo por iteración para el caso ***a*** en el que no se utilice la macrocoordenada para el cálculo de la distancia y por lo tanto se calculen todas las distancias entre todos los peces, será:

$$T_{Iter}^{P} a \approx K\left(\frac{\sum_{I} g_i}{P}\left(\frac{\sum_{I} g_i}{P} - 1\right)\right) \tag{62}$$

Esta aproximación del tiempo por iteración puede ser refinada puesto que el total de individuos por submódulo $\frac{\sum_{I} g_i}{P}$ está compuesto por los peces de los Gi grupos (cuyos individuos interactúan entre sí).

Nuevamente, considerando una distribución de individuos equiprobable en el *módulo de cómputo* se tiene el caso ***b*** en el que el tiempo aproximado por iteración teniendo en cuenta la interacción entre los individuos de mismo bloque solamente, se lo puede considerar proporcional a:

$$T_{Iter}^{P} b \approx K\left[\sum_{I}\left(\frac{g_i}{P}\left(\frac{g_i}{P} - 1\right)\right)\right] \tag{63}$$

Para este último caso el tiempo está determinado por la interacción de individuos del mismo grupo.

A continuación se hará un ejemplo para apreciar las diferencias de tiempo entre las últimas fórmulas de tiempo mostradas.

Sean:

P= 10, 100 y 200

$I = 5$

g_i = 1000/1000/2000/6000/10000

1) $P = 10$

$$T_{Iter}^{10}a = K\left(\frac{20000}{10}\left(\frac{20000}{10}-1\right)\right) = K3998000$$

$$T_{Iter}^{10}b = K\left[(100*99)+(100*99)+(200*199)+(600*599)+(1000*999)\right] = K1418000$$

2) $P = 100$

$$T_{Iter}^{100}a = K\left(\frac{20000}{100}\left(\frac{20000}{100}-1\right)\right) = K39800$$

$$T_{Iter}^{100}b = K\left[(10*9)+(10*9)+(20*19)+(60*59)+(100*99)\right] = K14000$$

3) $P = 200$

$$T_{Iter}^{200}a = K\left(\frac{20000}{200}\left(\frac{20000}{200}-1\right)\right) = K9900$$

$$T_{Iter}^{200}b = K\left[(5*4)+(5*4)+(10*9)+(30*29)+(50*49)\right] = K3450$$

Como se ve en las aproximaciones de los tiempos obtenidos, para una misma cantidad de procesadores la discriminación de los peces por el *Gi* genera una ganancia de tiempo mayor que dos.

Capítulo 5
Experimentación

5.1. Introducción

El objetivo del presente capítulo se basa en la experimentación de las ideas presentadas en los capítulos previos. En el capítulo 3 se describió un modelo bidimensional de Fish School y se hicieron las generalizaciones necesarias para llevar este modelo al espacio tridimensional. En el capítulo 4 se realizó un modelo analítico del simulador que permite la predicción de diversas variables del simulador que son útiles para conocer el tiempo de simulación y los recursos necesarios para una simulación determinada. En el mismo capítulo también se ha hecho una propuesta acerca de la forma en la que se puede tratar la cuestión de la distribución de la carga. Un conjunto de experimentos muestran el correcto funcionamiento del modelo implementado y la viabilidad de la propuesta realizada.

5.2. Modelo biológico Fish School

El modelo utilizado para la implementación del simulador es, como ya se comentó en los capítulos anteriores, un modelo orientado al individuo. Este tipo de modelos permite obtener el comportamiento de un banco de peces a partir de la interacción de los peces que lo componen.

El correcto funcionamiento del modelo puede ser analizado a dos niveles. Por un lado a nivel del comportamiento grupal, y por el otro lado, a nivel de la interacción de los peces. Estas dos cuestiones, que serán estudiadas en el presente apartado, han sido inspiradas en las

experimentaciones realizadas por los autores del modelo, Huth y Wissel [37] [36], para corroborar el correcto funcionamiento.

Para el análisis a nivel grupal del modelo se comparó el comportamiento típico de un banco de peces reales con los resultados obtenidos de la simulación para una situación análoga utilizando la herramienta desarrollada. El escenario escogido es el que se describe en [37] [36] y que dice que cuando dos bancos de peces se encuentran en aguas abiertas, la nueva dirección de movimiento final se aproxima al vector resultante de la unión de la trayectoria de los dos bancos representados como vectores.

La experimentación se realizó utilizando dos grupos de ocho peces cada uno (la misma cantidad utilizada en el modelo de Huth y Wissel). Los resultados de esta prueba se pueden ver en la figura 43. En esta gráfica se han dibujado todos los puntos por los que pasan los individuos del sistema. Como se observa, luego de pasado un tiempo de la unión de los dos grupos, el banco de peces resultante deja de desplazarse en la dirección que se aproxima a la suma de los vectores de desplazamiento de los grupos. Esto se debe a la aleatoriedad en la determinación de la nueva posición (para mas detalles ver el capítulo 3 y 4).

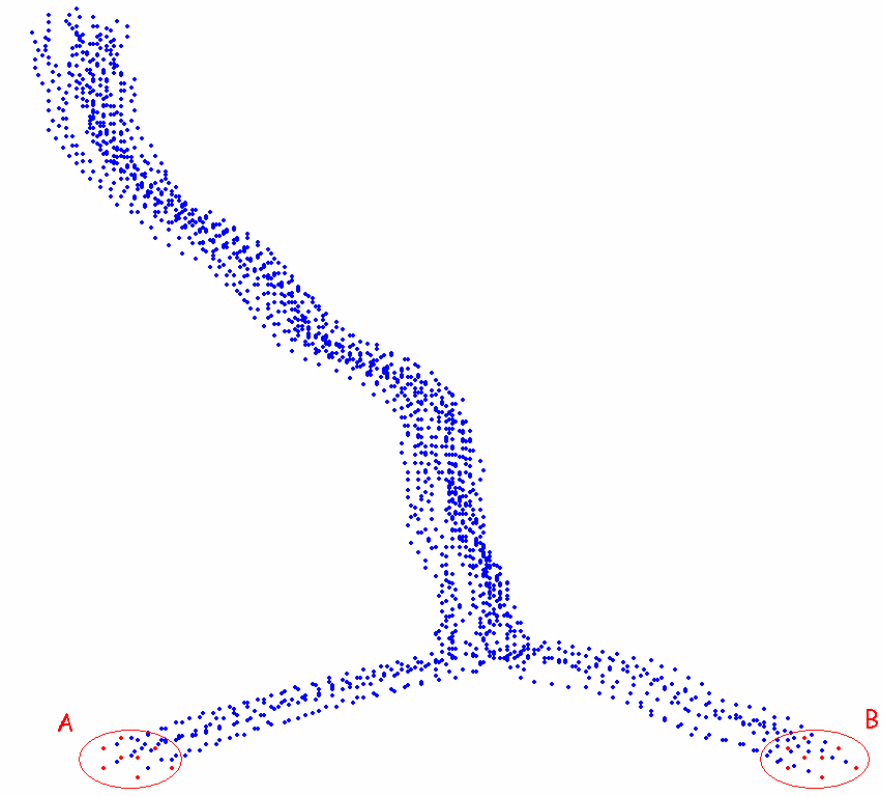


Figura 43: Unión de dos bancos de peces

El mismo experimento se repitió pero eliminando la aleatoriedad comentada anteriormente. Con esto, el simulador en vez de elegir entre algunos de los puntos más cercanos fijados por la discretización, se realiza un truncamiento y por lo tanto, opta por la parte entera de la coordenada exacta calculada. Los resultados de esta experimentación son los que se pueden observar en la figura 44. El movimiento después de la unión de los grupos pasa por una período transitorio y luego su desplazamiento es en el mismo sentido que el vector resultante de la unión de dos vectores que representarían el desplazamiento de los dos grupos.

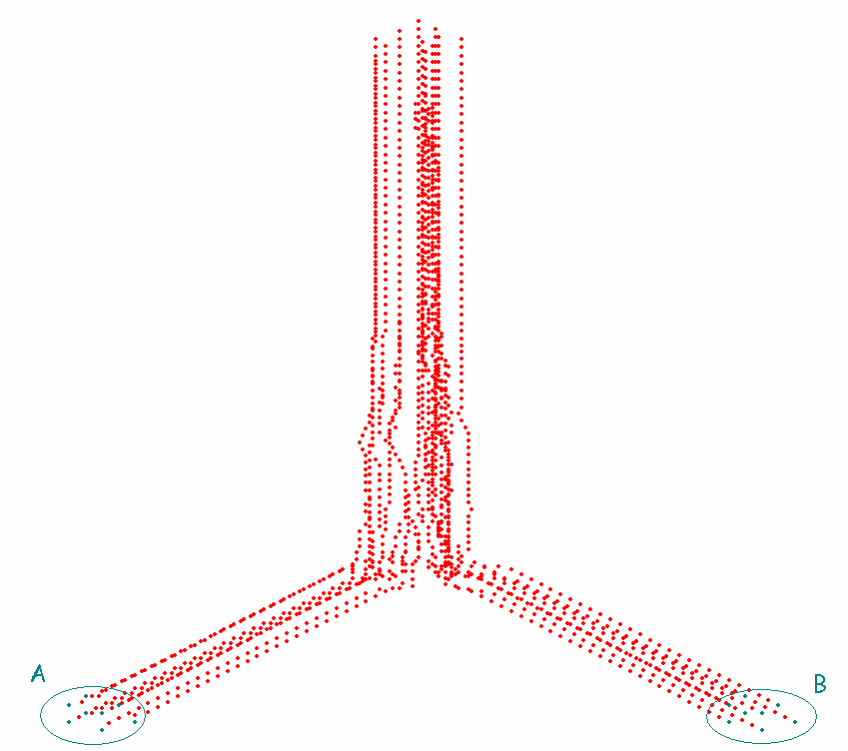


Figura 44: Unión de dos bancos de peces sin aleatoriedad

Estos experimentos permiten verificar dos comportamientos típicos de los peces. El primero es el movimiento en grupo de los individuos. El segundo es la formación de un grupo como resultado del encuentro de dos bancos de peces.

Las pruebas realizadas anteriormente no son suficientes para garantizar que el comportamiento simulado de un banco de peces es correcto, existen otros factores que deben ser analizados.

Un banco de peces de una determinada especie se caracteriza por la proximidad (cohesión) entre los individuos y por el grado de paralelismo de sus movimientos respecto al movimiento

del grupo. La primer característica puede ser cuantificada con la distancia del vecino más cercano (Nearest neighbour Distance - NND) y la segunda con la polarización.

La NND se define como el promedio de la distancia de cada pez con su vecino más cercano. La polarización es la intensidad de orientación paralela en el banco de peces y está definida por el promedio del ángulo de desviación de cada pez con el promedio de la dirección de movimiento del grupo.

De acuerdo a la especie, los valores de NND y polarización varían. Estos dos valores se hallan estrechamente ligados a los valores de los tres radios de influencia y a la relación entre ellos. Por ejemplo, un banco de caballas muestra una alta orientación paralela y una alta cohesión. Los valores para esta especie son: polarización=8 – 12º y NND=0,5BL.

Los experimentos se realizaron con los siguientes valores de radios:

- R_1=1BL, R_2=4BL, R_3=6BL

Los valores obtenidos de polarización y NND para estos radios son:

- Polarización = 10º
- NND = 0,8BL

Se puede concluir que los valores de los radios utilizados en la experimentación corresponden a una especie que tiene una elevada polarización y cohesión como es el caso de un banco de caballas [36].

5.3. Modelo analítico y prestaciones del simulador

En el capítulo anterior se ha analizado la estructura del simulador y se obtuvieron fórmulas que representan a cada una de las partes que componen el código del simulador. Estas fórmulas permiten predecir o estimar el tiempo que tardará en realizarse una iteración de acuerdo a los recursos disponibles y a las características del sistema a simular.

Las experimentaciones se han realizado utilizando un cluster homogéneo con las siguientes características hardware y de sistema operativo:

- Cantidad de procesadores: 16

- Procesadores: Pentium IV 1,8GHz
- Memoria: 512Mbytes DDR266
- Red de interconexión: Fast Ethernet
- Sistema operativo: SuSe LINUX 8

El simulador se instrumentó de manera tal que se pueda conocer el tiempo utilizado en cada una de las partes. De esta forma se obtienen las distintas constantes que son parte de la ecuación para determinar el tiempo por iteración. La ecuación (57) con los valores de las constantes es la siguiente:

$$T_i = \frac{n^2 R_3}{PL} 3{,}024 \times 10^{-7} + \frac{n^2}{P^2} 1.37 \times 10^{-7} + \frac{n^2 {R_3}^2}{L^2} 2{,}32 \times 10^{-5} - \frac{n^3 {R_3}^3}{L^3} 1{,}24 \times 10^{-9} - \frac{nR_3}{L} 0{,}015 + 6{,}106$$

Las otras dos ecuaciones que sirven para el cálculo de la cantidad de procesadores necesarios o la cantidad de individuos que se pueden simular en determinada condición, son obtenidas también en base a las constantes mencionadas anteriormente.

En muchos sistemas distribuidos, como es el caso del simulador desarrollado en la presente tesis, existe una medida que presenta gran interés que es la escalabilidad. Esta medida se planteará de dos formas.

La primera consiste en conocer cuanto se puede reducir el tiempo de simulación de un sistema aumentando la cantidad de procesadores utilizados. Con esta escalabilidad lo que se obtiene es un límite para el cual, un aumento de los recursos computacionales no genera una reducción de tiempo que justifique el costo de agregar más procesadores.

La segunda forma en la que se puede plantear la escalabilidad consiste en determinar cuantos individuos y procesadores son necesarios para que una iteración dure un tiempo prefijado. La escalabilidad, en este caso, es de la cantidad de individuos, es decir, lo que se escala es la complejidad del problema mediante el incremento de los peces.

La escalabilidad de procesadores se realizó bajo la metodología que se explica a continuación. Para un determinado espacio de simulación y cantidad de individuos, se ejecutó una simulación con 16 procesadores. A continuación se simuló la mitad del espacio y de peces,

con 8 procesadores. De la misma forma se realizó una simulación con 4 procesadores. Los tiempos obtenidos fueron similares. Así se puede apreciar el correcto funcionamiento del simulador en cuanto a escalabilidad.

De esta forma también puede simularse y conocerse el tiempo consumido para la misma cantidad de peces que se simuló con 16 procesadores pero haciéndolo ahora con 8 y 4 procesadores. Esta metodología será la utilizada para escalar a valores superiores de 16 procesadores.

Las simulaciones para el análisis de la escalabilidad se realizaron utilizando tres escenarios distintos de simulación, E1, E2 y E3. Las características de ellos son las que se muestran en la tabla 2.Como se observa, los tres escenarios tienen el mismo volumen, lo que cambia entre cada uno de ellos es la densidad de individuos (o cantidad de individuos). La selección de estos escenarios permite analizar como se pueden obtener beneficios temporales en la distribución del modelo de simulación para poblaciones con grandes cantidades de individuos y la escalabilidad de procesadores.

La distribución de los peces en los distintos espacios de simulación se realiza en forma aleatoria a lo largo de todo el espacio de simulación. De esta forma se consigue que cada proceso lógico simule la misma cantidad de individuos en media y por lo tanto el sistema se encuentra balanceado.

Tabla 2: Escenarios de Simulación

	E1	**E2**	**E3**
Cantidad de individuos	640000	512000	256000
Dimensiones del mundo de simulación	25600x500x500	25600x500x500	25600x500x500

Los experimentos se realizaron comenzando con la distribución del mundo de simulación en 8 procesadores. Luego, aplicando la metodología comentada anteriormente se continuó escalando hasta llegar a 512 procesadores. En la tabla 3 se muestran los tiempo de simulación obtenidos.

Tabla 3: Tiempos de simulación [Segundos]

Procesadores	640000	512000	256000
8	930,72	596,77	150,01
16	242,72	157,47	39,33
32	68,65	43,81	11,97
64	23,97	16,05	5,31
128	13,35	10,05	4,04
256	10,25	8,10	3,70
320	10,03	7,63	3,66
512	8,00	5,68	3,01

A medida que el sistema aumenta su número de procesadores, el tamaño de los procesos lógicos disminuye su ancho. La cantidad de individuos por bloques también lo hace como lo muestra la tabla 4.

Tabla 4: Cantidad de individuos y tamaño de los procesos lógicos

Procesadores	640000		512000		256000	
	Dimensiones del LP	Cantidad de Individuos	Dimensiones del LP	Cantidad de Individuos	Dimensiones del LP	Cantidad de Individuos
8	3200x500x500	80000	3200x500x500	640000	3200x500x500	32000
16	1600x500x500	40000	1600x500x500	32000	1600x500x500	16000
32	800x500x500	20000	800x500x500	16000	800x500x500	8000
64	400x500x500	10000	400x500x500	8000	400x500x500	4000
128	200x500x500	5000	200x500x500	4000	200x500x500	2000
256	100x500x500	2500	100x500x500	2000	100x500x500	1000
320	80x500x500	2000	80x500x500	1600	80x500x500	800
512	50x500x500	1250	50x500x500	1000	50x500x500	500

Los tiempos de la tabla 3 se graficaron en la figura 45 donde se puede ver el tiempo por iteración versus la cantidad de procesadores. Como parámetro de las curvas se utilizó el número de individuos en el espacio de simulación.

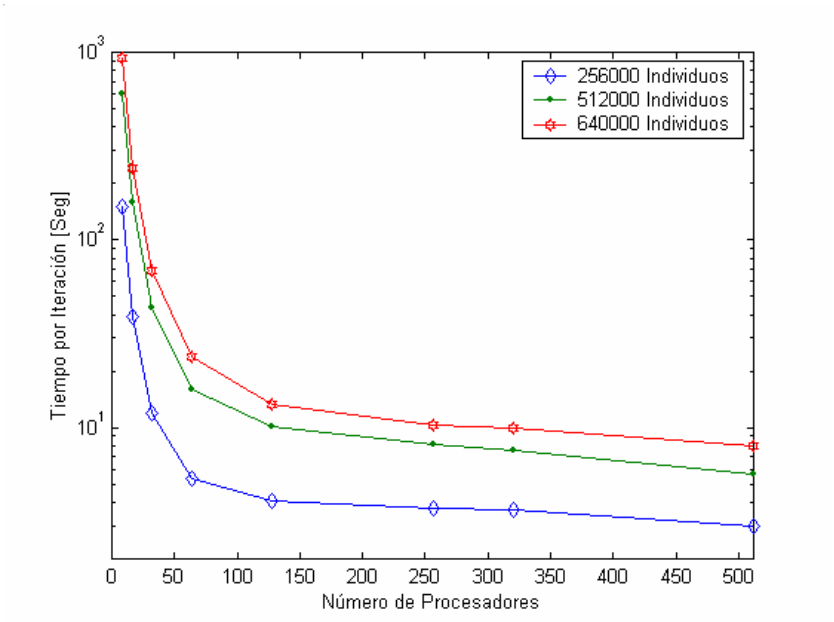


Figura 45: Tiempo por iteración [Segundos]

En la figura 45 se aprecia que a partir de 128 procesadores, la ganancia de tiempo que se obtiene con la escalabilidad no es considerable. En este caso el rendimiento del sistema no justifica el incremento de los recursos.

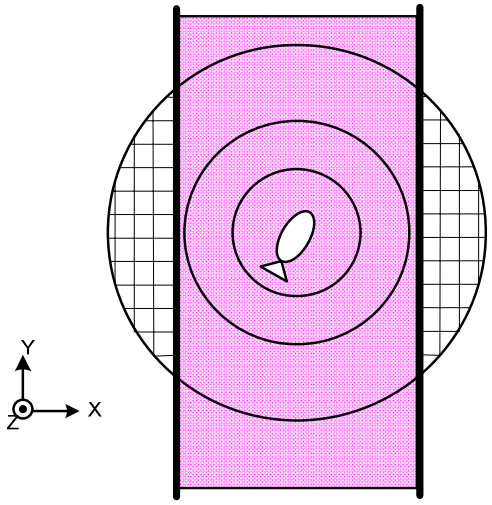


Figura 46: Caso particular de ancho de los procesos lógicos

Los valores escogidos para la escalabilidad de procesadores presentan una cierta característica, y es que para las cantidades más altas de procesadores no se continuó con

valores que fueran potencias de dos. Esto se debe a que para 512 procesadores, el ancho del proceso lógico es menor que el diámetro determinado por el máximo radio de influencia (figura 46). Igualmente se han hecho las simulaciones para ese ancho de proceso lógico para analizar si el modelo del simulador tiene validez para este caso.

En la figura 47 se puede observar con más detalle el tiempo por iteración para simulaciones con más de 64 procesadores. El comportamiento del simulador muestra una tendencia a mantener un valor constante de tiempo por iteración. El tiempo no decrece en forma superlineal como sucede para cantidades menores de 128 procesadores.

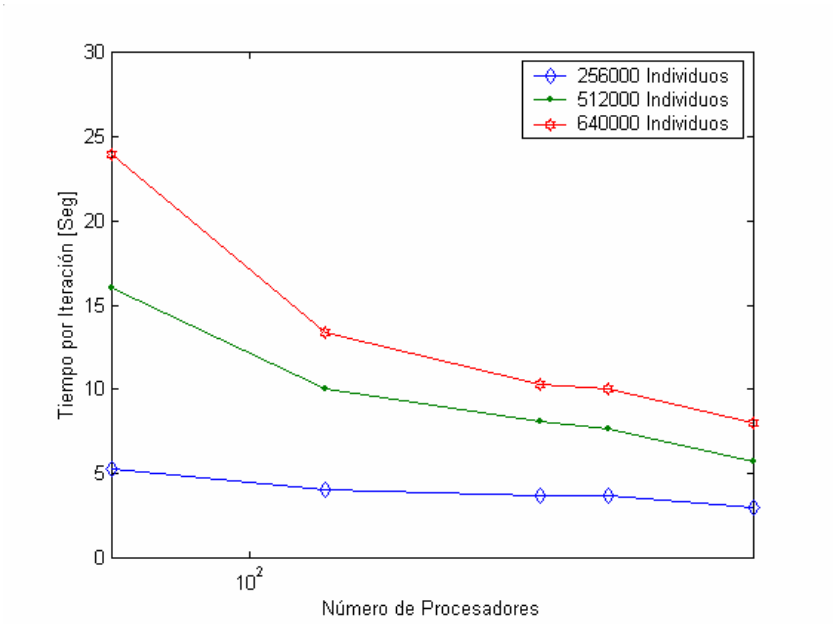


Figura 47: Tiempo por iteración para más de 64 procesadores

Para analizar el razón del comportamiento mencionado anteriormente en el que el tiempo de simulación no disminuye aunque se agreguen más procesadores, en la

tabla 5 se reproducen los cuatro tiempos que forman parte del tiempo por iteración para el escenario de 640000 individuos y todo el rango de procesadores experimentados. El tiempo de la segunda parte se lo ha desglosado para mostrar:

- El tiempo de cómputo utilizado para contestar las preguntas y procesar la información de los mensajes recibidos ($T2_R$)
- El tiempo de comunicación y sincronismo implícito ($T2_C$) (para mas detalles ver capitulo 4).

Tabla 5: Tiempos para el escenario E1 [Segundos]

Procesadores	T1	$T2_R$	$T2_C$	T3	T4	T_{Total}
8	830,8	4,9	11,7	14,31	69	930,71
16	207,55	2,408	8,72	7,09	16,95	242,718
32	50,8	1,22	9,04	3,62	3,96	68,64
64	12	0,61	8,707	1,85	0,8	23,967
128	2,7	0,3	9,194	0,972	0,173	13,339
256	0,501	0,154	8,966	0,549	0,046	10,246
512	0,057	0,069	7,57	0,28	0,014	7,99

La figura 48 muestra en forma de curvas los datos de la

tabla 5. Como se puede apreciar, los tiempos T_1, $T_{2Cómputo}$, T_3 y T_4 que son derivados de algoritmos del tipo $O(N^2)$ siguen el comportamiento esperado cuando los individuos en cada proceso lógico disminuye. El tiempo de comunicación permanece prácticamente constante para las distintas cantidades de procesadores utilizados. Esto se debe a que la cantidad de individuos que consultan a los bloques vecinos es la misma puesto que la densidad de individuos es la misma.

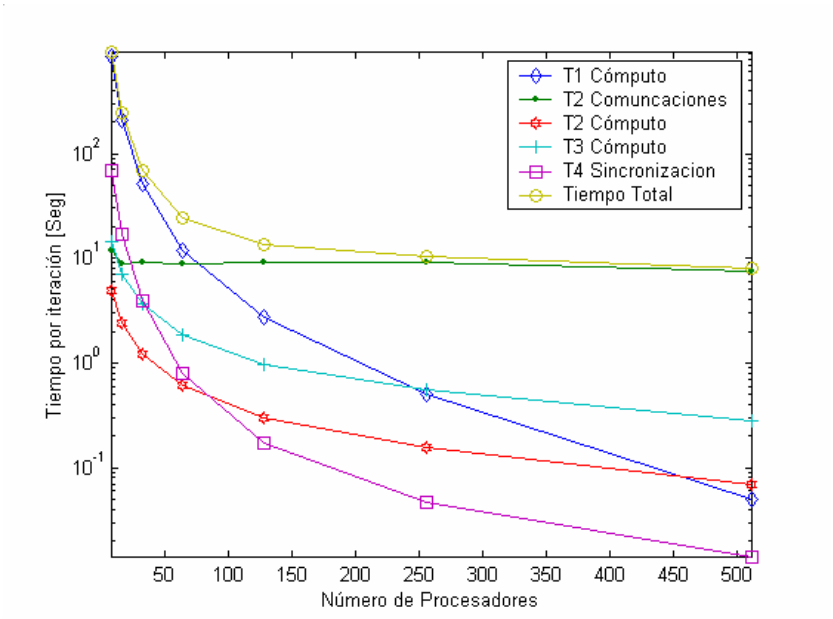


Figura 48: Tiempos por Iteración para el escenario E1

Para una visualización más clara de los tiempos para menos de 32 procesadores se ha hecho una ampliación del grafico en la zona correspondiente (figura 49).

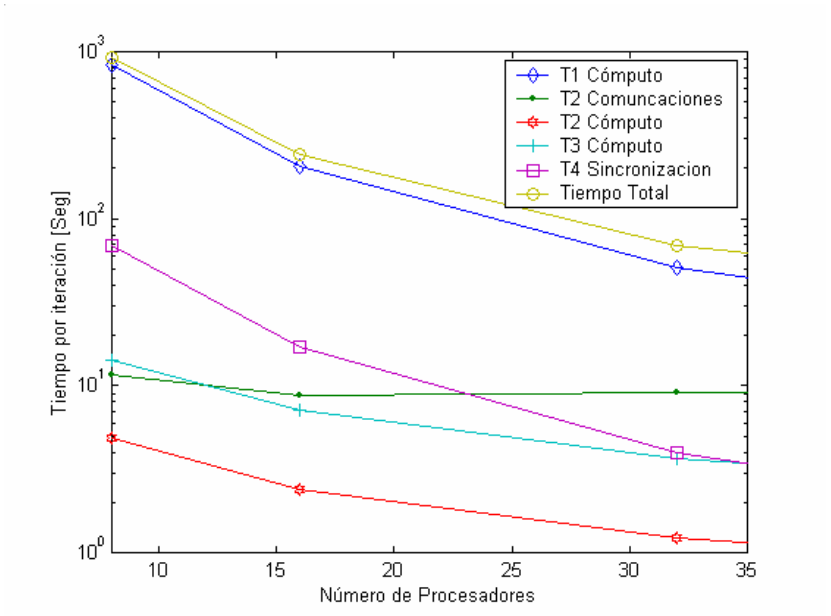


Figura 49: Tiempos por Iteración para el escenario E1 para el rango de 8-32 procesadores

En las figura 48 y figura 49 queda reflejada la influencia que tienen las comunicaciones cuando las etapas de cómputo deben resolverse con menos individuos y por ende su tiempo de cómputo disminuye exponencialmente. EL tiempo de comunicación marca un umbral mínimo para el tiempo por iteración.

En la figura 50 se muestra el porcentaje de tiempo utilizado por cada parte cuando se escala de 8 procesadores a 512 simulando 640000 individuos. En esta gráfica también puede corroborarse que para más de 128 procesadores, las comunicaciones son las que prevalecen frente al cómputo y hacen que la disminución de tiempo tienda a estabilizarse en torno al tiempo utilizado por las comunicaciones.

Otra de las mediciones que ayudan a realizar una mejor comprensión del funcionamiento del simulador en lo concerniente a las prestaciones es el SpeedUp [59] [75]. Esta medida es calculada como la relación entre el tiempo de ejecución serie y el paralelo:

$$SpeedUp = \frac{T_{Serie}}{T_{Paralelo}} \tag{64}$$

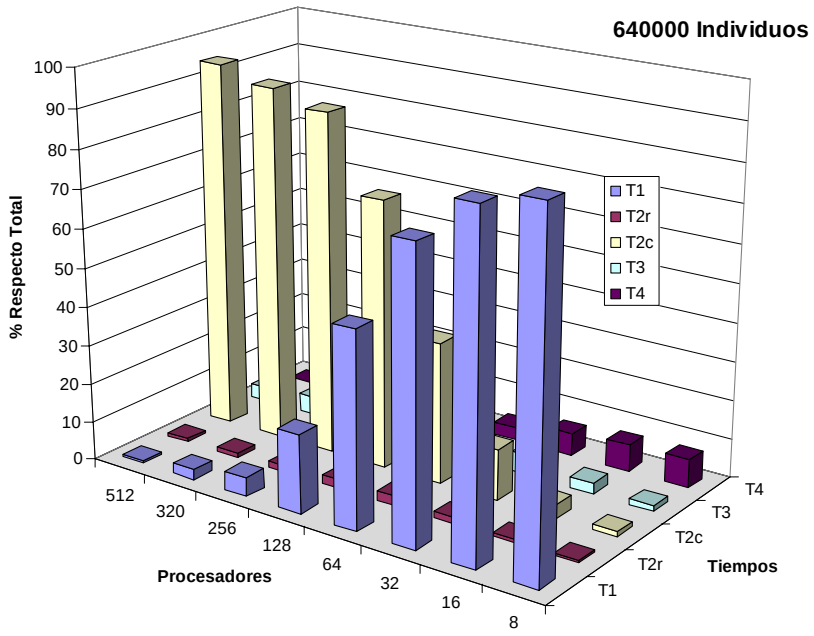


Figura 50: Detalle de los tiempos utilizados

Si el escalado del sistema es ideal, por cada procesador que se agregue, la relación de tiempos también lo hará de la misma forma. Esto implica que si se grafica el SpeedUp versus el número de procesadores se obtiene una recta con pendiente uno. Dicha recta se denominará SpeedUp lineal.

En general, cuando los sistemas escalan tienen como límite superior de SpeedUp al lineal. En el caso del simulador en estudio, se tienen casos de superlinealidad. Esto implica que el

SpeedUp obtenido es superior al lineal. El motivo de este comportamiento es la reducción de la complejidad cuando se incrementa la cantidad de procesadores.

Puesto que los tiempos de simulación son función del grado de complejidad a simular, como primera aproximación, el SpeedUp del simulador puede ser expresado:

$$SpeedUp = \frac{KN^2}{K(N/P)^2} = P^2 \tag{65}$$

P^2 es el valor máximo que puede tomar el SpeedUp para el simulador y se lo denominará SpeedUp ideal.

El cálculo hecho no tiene en cuenta el peso de las comunicaciones en el tiempo de simulación para cantidades grandes de procesadores. Este efecto se ha comentado anteriormente y para las experimentaciones realizadas tienen importancia para valores superiores a los 128 procesadores.

En la figura 51 pueden visualizarse los valores obtenidos de SpeedUp. Como se ha dicho, el peso de las comunicaciones frente al cómputo produce que la ganancia se aleje en forma considerable de la ideal a partir de los 128 procesadores. Este efecto se manifiesta por la disminución de la pendiente de las curvas de SpeedUp obtenidas.

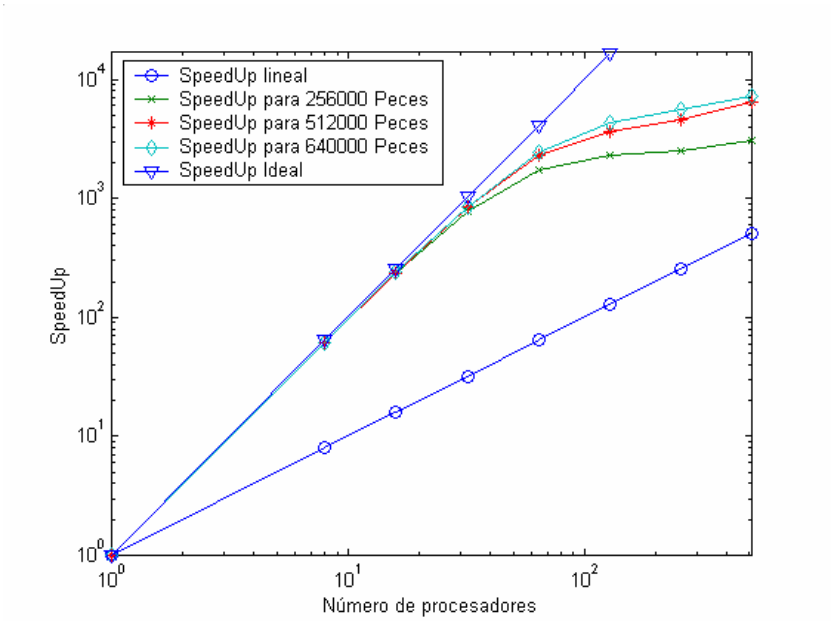


Figura 51: SpeedUp

En base a las experimentaciones realizadas se puede verificar que el sistema escala correctamente y que la limitación en la escalabilidad de procesadores es debida a dos factores: las comunicaciones, como se acaba de analizar y al ancho del proceso lógico que no puede ser inferior a dos veces el radio máximo de influencia (R_3).

Las valores obtenidos con la formula para la predicción del tiempo de simulación se compararon con los datos experimentales. En la figura 52 se puede apreciar la forma en la que la estimación del tiempo se aproxima al tiempo real.

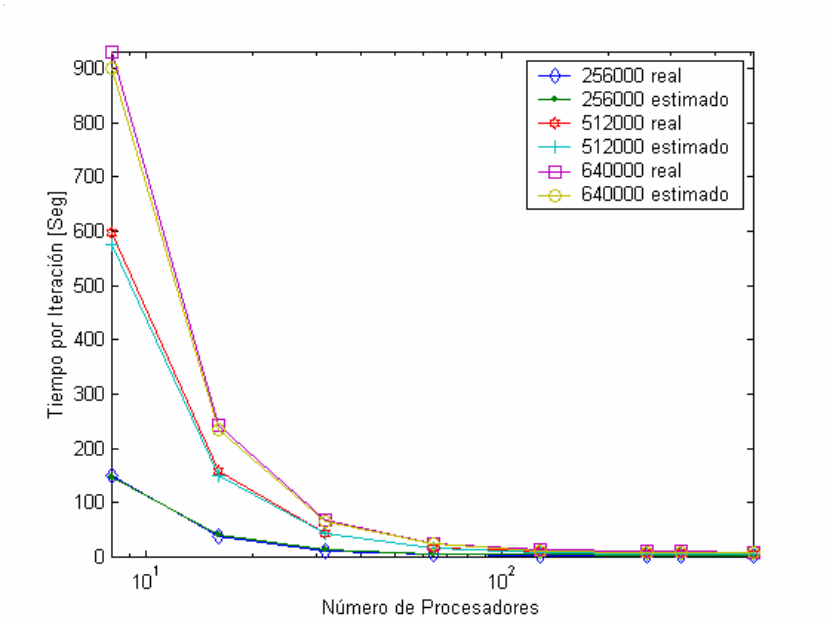


Figura 52: Tiempo real y estimado [Segundos]

La figura 53 y la figura 54 son ampliaciones de la gráfica que representa el tiempo por iteración real y estimado para los rangos de 8 a 64 y de 64 a 512 procesadores. La diferencia entre estos dos tiempos hasta los 128 procesadores es muy reducida.

En la figura 54 se observa que para el caso de 512 procesadores presenta una modificación de la tendencia en el tiempo por iteración. Por otro lado las curvas de tiempo estimado no muestran una variación tan marcada en su derivada. La forma en la que varía el tiempo real para 512 procesadores es otro de las pruebas que demuestran que el simulador no responde en forma adecuada en esas condiciones.

Como se desprende de la figura 53 y figura 54 el error cometido en la aproximación es reducido. Si se tiene en cuenta el rango de 8 a 320 procesadores (rango en el cual es válido el modelo de simulación), el error promedio es de ±4,25% (o 8,74% en valor absoluto). Para 512

procesadores, el error no es tenido en cuenta ya que el modelo y el simulador no contempla el caso en el que los procesos lógicos sean más angostos que dos veces el R_3.

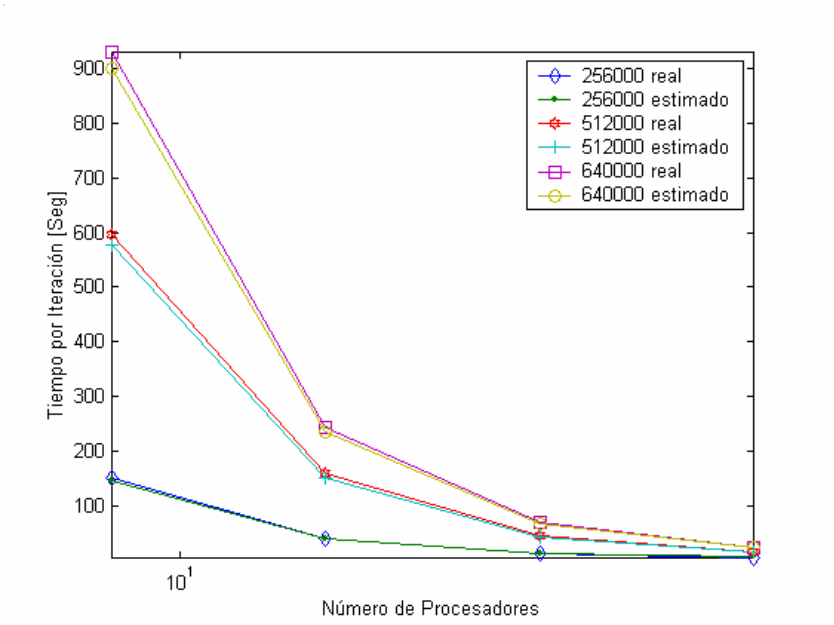


Figura 53: Tiempo real y estimado de 8 a 64 procesadores [Segundos]

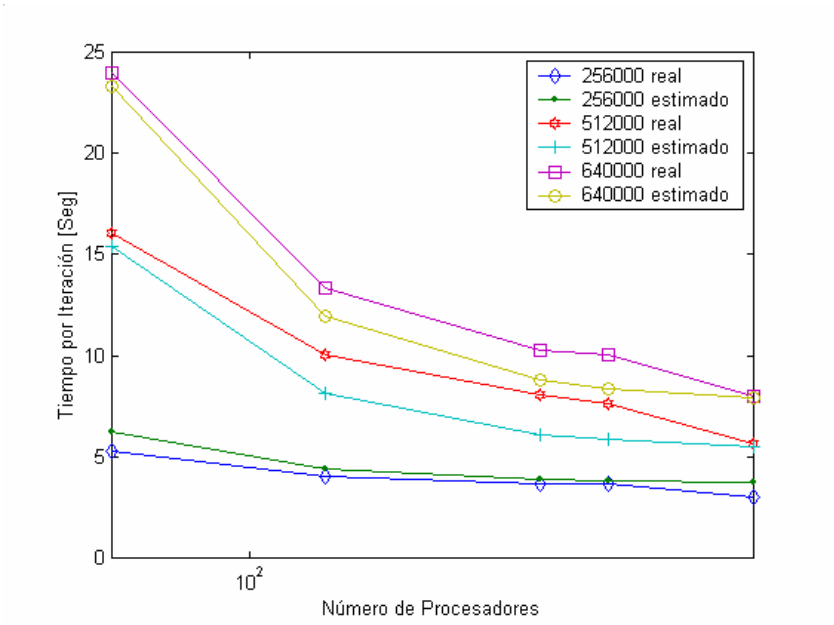


Figura 54: Tiempo real y estimado para más de 64 procesadores [Segundos]

Como se comentó en el comienzo del presente apartado, se analizarán dos tipos de escalabilidad. La de procesadores, como se acaba de ver, permite conocer hasta que cantidad de procesadores se pueden agregar al sistema para resolver un problema determinado, es decir, una cantidad fija de individuos y un espacio de simulación determinado. La segunda de

las escalabilidades que se analizará es la escalabilidad del problema. En este caso, lo que se busca es saber hasta cuantos individuos se pueden simular para un espacio y número de procesadores fijos.

Para la experimentación de escalabilidad del problema, se ha aplicado la formula (59) obtenida en el capítulo 4. La variación de procesadores ha sido entre 8 y 256. El espacio de simulación se mantuvo constante para estas pruebas y fue calculado de manera tal que el ancho de los procesos lógicos sea siempre mayor que dos veces el radio máximo de influencia de los peces. Los tiempos utilizados para este caso son: 10, 20, 40, 80, 160 y 320 segundos.

En la tabla 6 se pueden apreciar las cantidades de individuos que satisfacen las condiciones de tiempo y cantidad de procesadores impuestas para esta experimentación.

Tabla 6: Escalabilidad en la complejidad del problema [Número de Individuos]

	Número de Procesadores					
Tiempo por Iteración [Seg/Iteración]	**8**	**16**	**32**	**64**	**128**	**256**
10	43939	100000	200000	400000	830000	1433000
20	81761	200000	300000	680000	1300000	2101000
40	126930	300000	500000	1020000	1880000	2953000
80	186760	400000	700000	1460000	2670000	4116000
160	268910	500000	1100000	2080000	3750000	5756000
320	383460	800000	1500000	2930000	5270000	8115000

En la figura 55 se observan los valores de la tabla 6 en una gráfica de número de procesadores versus cantidad de procesadores. Puesto que estos valores estimados son obtenidos de la fórmula (59), existirán variaciones en el tiempo de simulación cuando se apliquen estos valores.

En la figura 56 se muestran los tiempos prefijados para la experimentación y los tiempos obtenidos utilizando las cantidades de individuos de la tabla 6. Como se puede apreciar, las

estimaciones son mejores para los valores en los cuales es mayor la cantidad de procesadores y de tiempo de simulación.

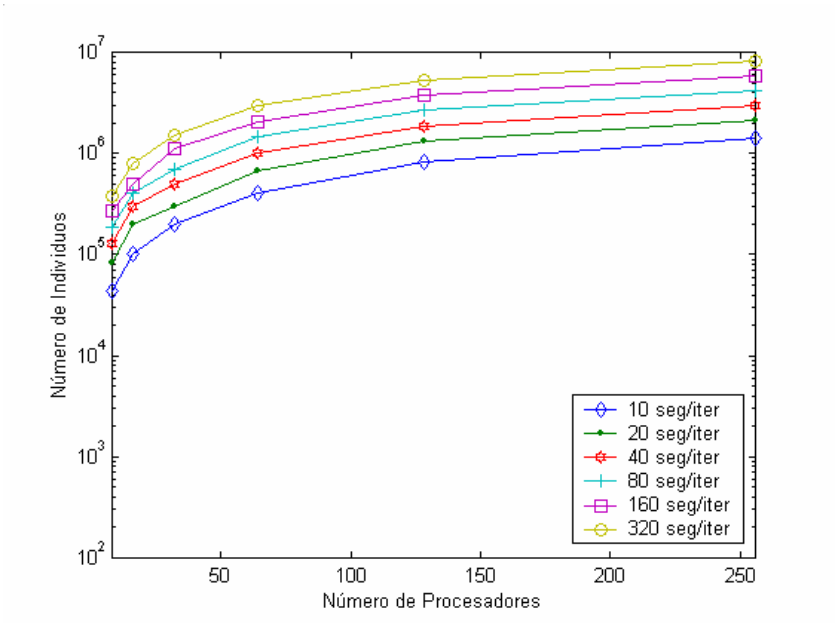


Figura 55: Escalabilidad en la complejidad del problema

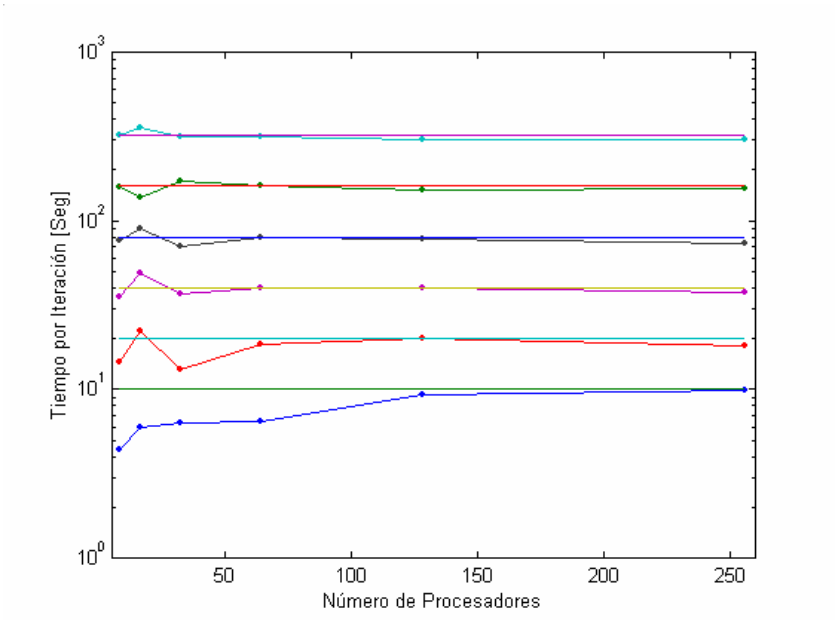


Figura 56: Tiempos preestablecidos y estimados

Debido a que en esta experimentación se ha mantenido el espacio de simulación constante, hay combinaciones calculadas en las que la densidad de individuos es muy pequeña. Esta disminución de la densidad genera una desviación mayor entre el valor de tiempo preestablecido y el tiempo experimentado con los valores de la tabla 6.

La figura 57 y figura 58 muestran la densidad y el error absoluto respectivamente para la experimentación de escalabilidad del problema. La diferencia entre el tiempo supuesto y el que se obtiene con los individuos estimados, se encuentran en el mayor número de casos por debajo del 10%. Si se observa con detenimiento, la figura de densidades y de error se puede concluir que el mayor error se encuentra en las combinaciones de procesadores y tiempos en los que la densidad es baja.

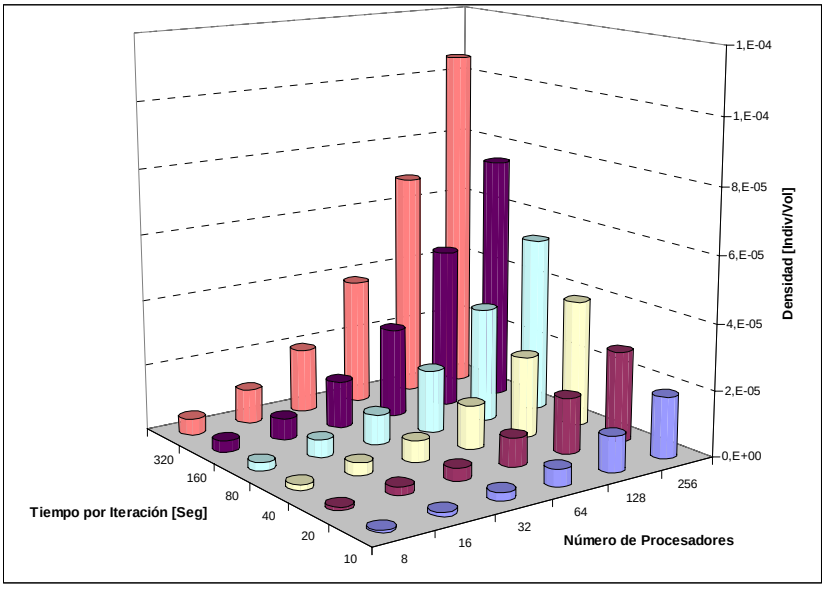


Figura 57: Densidad

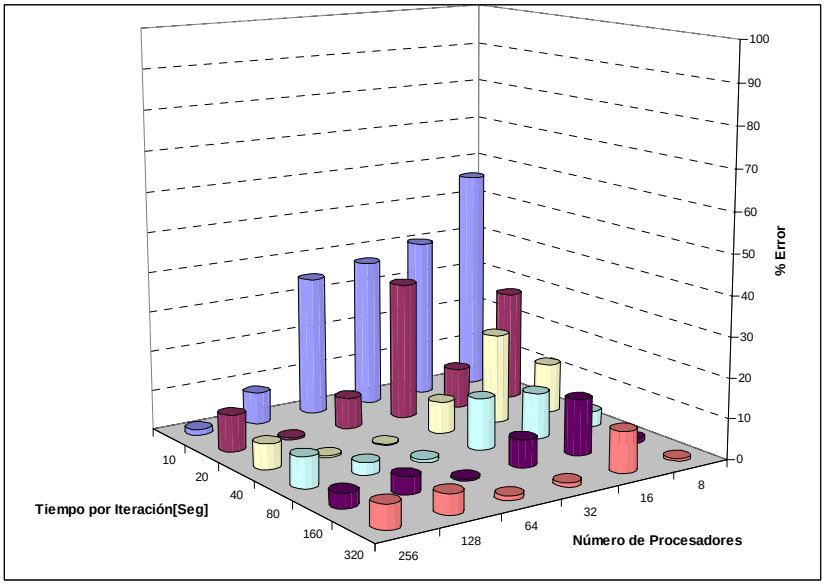


Figura 58: Error absoluto

Esta situación se debe a que en simulaciones en las que la densidad es demasiada reducida, el modelo analítico no responde adecuadamente puesto que la interacción entre individuos no se encuentra garantizada. Esto quiere decir que si la densidad tiene valores muy bajos, la

separación entre individuos puede ser tal que la separación entre los peces sea mayor que el máximo radio (R_3).

5.4. Conclusión de las experimentaciones realizadas para demostrar la viabilidad de la propuesta

En la experimentación con grandes cantidades de individuos como la realizada en el apartado anterior, lo que se persigue es demostrar que la propuesta de otra forma de distribuir el sistema es viable.

Si bien las simulaciones realizadas no son totalmente representativas del comportamiento de un banco de peces, son igualmente útiles para conocer la carga computacional que implica la simulación de grandes cantidades de individuos.

El módulo computacional contendría todo el espacio de simulación. Esto quiere decir que un procesador podría contener parte (o todo) de uno o varios bancos de peces. En las simulaciones realizadas con el método clásico existe el problema de los bordes externos del espacio de simulación. Los peces que cruzan estas fronteras vuelven a ser incorporados por la cara opuesta. En estas circunstancias estos individuos dejan de tener a los vecinos que estaban determinando su movimiento y el grupo ya no responderá como lo hace en la naturaleza.

El motivo por el cual se ha utilizado igualmente una distribución uniforme y aleatoria en todo el espacio de simulación es para emular la carga que se tendría aplicando la nueva distribución del sistema. Para este caso la salida de los peces por las caras laterales no implicaría ningún problema de discontinuidad. El individuo sería reincorporado pero ahora con una macrocoordenada distinta que identifica la posición del bloque al cual pertenece y que está mapeado en el módulo de cómputo.

El poder de cómputo utilizado en las experimentaciones es similar a las que se tendrían en una simulación aplicando la propuesta de distribución del espacio en la situación más desfavorable. Esto quiere decir que sería en el caso en el que no se utilizaran las macrocoordenadas. El no utilizarlas tiene como consecuencia la consulta de distancia (cálculo de la distancia euclidiana) de cada individuo con el resto de los que se encuentran en el mismo bloque y la generación de consultas al bloque vecino.

Los resultados obtenidos son útiles para tener una primera aproximación al tiempo máximo que se podría utilizar para una simulación en el caso en el que las macrocoordenadas no fueran tenidas en cuenta.

Capítulo 6
Conclusiones y Líneas abiertas

6.1. Conclusiones

Se ha demostrado que diversos ámbitos de las ciencias pueden valerse de herramientas de simulación para avanzar en sus conocimientos. El presente trabajo se basó en la metodología propuesta por [46], para la resolución de un problema enmarcado en la ciencia computacional. Para esto es necesario un modelo del problema a afrontar. Dicho modelo debería comenzar con el planteo físico que permite el paso al modelo analítico. Este paso entre los modelos fue realizado por Huth y Wiessel [37] para un espacio bidimensional. Un estudio riguroso del modelo y del comportamiento de los peces se transformó en el modelo tridimensional que se desarrolló y presentó en el capítulo 3. Para la verificación del modelo computacional se realizaron diversas pruebas que determinaron que este modelo se comporta de forma acorde al modelo matemático.

Si bien la verificación del modelo garantizó que la interacción de cada individuo con sus vecinos sea la correcta, el comportamiento global se validó con datos provenientes de la observación de bancos de peces. Los resultados obtenidos (figura 43) fueron satisfactorios, con lo que se puede concluir que el modelo computacional desarrollado es correcto para la representación de bancos de peces.

La simulación del modelo Fish School tridimensional desarrollado ha sido mediante la implementación de un simulador distribuido. El motivo de la utilización de esta estrategia fue

la elevada cantidad de tiempo que consume la simulación de este modelo orientado al individuo, en un simulador serie, cuando la cantidad de elementos a simular es elevada. En la tabla 7 se puede observar la ganancia de tiempo obtenida con la simulación distribuida.

Tabla 7: Tiempos de simulación serie y distribuida [Seg/Iteración]

		Individuos		
		640000	**512000**	**256000**
Procesadores	**1**	57717	36938	9233
	8	930,72	596,77	150,01
	16	242,72	157,47	39,33
	32	68,65	43,81	11,97
	64	23,97	16,05	5,31
	128	13,35	10,05	4,04
	256	10,25	8,10	3,70

El tiempo de simulación serie puede considerarse de complejidad $O(N^2)$(con N cantidad de individuos). En una primera aproximación, el tiempo para el simulador distribuido tiene una variación conforme a $O(N/P)^2$ (P número de procesadores). El SpeedUp ideal que se obtiene es de P^2 (superlineal), mientras que el lineal es de P. En la figura 51 se observa que los valores obtenidos de SpeedUp se hallan entre el lineal y el ideal. La escalabilidad que presenta el simulador es muy interesante, ya que en base a la distribución del espacio de simulación se puede reducir, por ejemplo, el tiempo de una iteración de 16 horas a 10 segundos utilizando 256 procesadores. La limitación que presenta la escalabilidad es la anchura que pueden tener los procesos lógicos (LP) simulados. Este ancho está relacionado con parámetros propios del modelo que influyen en la cantidad de mensajes de consulta enviados a los procesos lógicos vecinos. El incremento de los mensajes produce la limitación en la escalabilidad debido al tiempo consumido en las comunicaciones.

En una primera etapa del trabajo se ha realizado la implementación del modelo Fish School simplificado y distribuido, utilizando la librería de comunicaciones MPI en reemplazo de la

que se había utilizado anteriormente (PVM). Este cambio en las comunicaciones ha generado ganancias temporales que pueden encontrarse en:

> Mostaccio D., Suppi R. y Luque E., "Simulación Distribuida de Modelos Orientados al Individuo utilizando MPI". X Congreso Argentino de Ciencia de la Computación (CACIC 2004), Universidad de La Matanza, Argentina. Septiembre 2004.
>
> Mostaccio D., Suppi R. and Luque E., "Simulation of Ecologic System Using MPI". EuroPVM/MPI 2005, LNCS 2666, pp. 449-456. 2005

Otro de los aspectos en los que se trabajó es en el modelo analítico del simulador. Este permite al usuario la posibilidad de predecir la cantidad de procesadores, el tiempo de simulación y la cantidad de individuos que se pueden simular, para cada situación impuesta por el usuario. Estas predicciones se pueden realizar con un error menor al 15%. El modelo analítico es por tanto, una herramienta de gran utilidad que permite al usuario prever los recursos que necesitará para simular el escenario en estudio.

Los trabajos realizados ofrecen en su conjunto dos importantes aportes a la ciencia computacional, el primero es una forma de trabajo en la que se partió de un modelo bidimensional (que fue utilizado por Huth y Wiessel para simular pequeñas cantidades de individuos), y se llegó a un modelo computacional que fue implementado en un simulador distribuido, debido a los elevados requerimientos de poder cómputo necesarios.

El segundo de los aportes es la propuesta a la distribución tradicional del espacio de simulación. Con esta propuesta es posible incrementar el espacio y la cantidad de especies sin las limitaciones del único espacio de simulación que se tenía anteriormente. Además el problema del balanceo de carga tenderá a reducirse con la aplicación de esta propuesta permitiendo así un rendimiento mayor del sistema paralelo utilizado.

Los trabajos que motivaron a la propuesta realizada del mapeo del espacio total de simulación a un solo módulo de cómputo son:

> Mostaccio D., Suppi R. and Luque E., "Using Distributed Events Simulation for Individual Oriented Models". Internacional Mediterranean Multimodeling Multiconference (IM3). ISBN 2-9520712-3-3, pp. 105-110. 2005.

Mostaccio D., Suppi R. and Luque E., "Distributed Simulation of Ecologic Systems". XVI Jornadas de Paralelismo. Thomson. ISBN 84-9732-430-7, pp. 671-676. 2005.

Mostaccio D., Dalforno Ch., Suppi R. and Luque E., "Distributed Simulation of Large-Scale Individual Oriented Models". Journal of Computer Science & Technology. ISSN 1666-6038 Vol 6 pp. 59-65. 2006

La unión de estos dos aportes ofrece a los biólogos interesados en el estudio del comportamiento de bancos de peces, una herramienta que les facilita comprobar las reacciones de los individuos mediante la modificación de los parámetros característicos de los peces.

6.2. Líneas abiertas

Los trabajos realizados generaron la base para el desarrollo e implementación de diversas ideas entre las cuales se pueden encontrar las que se describen a continuación.

La implementación del módulo de cómputo que permitirá balancear la carga del sistema cuando el o los bancos de peces se desplacen. Con esto se podrán simular espacios en los que convivan diversas especies, lo cual se asemeja más al comportamiento existente en la naturaleza.

Los nuevos elementos que se han comentado tienen la finalidad de generar escenarios de simulación más cercanos a la realidad, es decir, que la simulación total sea el producto de la simulación de diferentes especies y por lo tanto lo que se esté simulando sea una parte de un ecosistema. Este incremento de individuos y elementos externos tiene como agregado un aumento en la complejidad de los modelos simulados y por lo tanto la potencia de cómputo requerida. La distribución del modelo para la disminución del tiempo de cómputo, es por tanto indispensable para obtener simulaciones en tiempo reducidos.

Además de incrementar la cantidad de especies, otra forma de conseguir una simulación de un sistema que se asemeje más a uno real, es la incorporación de obstáculos. Estos representarían por ejemplo el fondo del mar o la vegetación que se puede encontrar.

Otro factor que aproximaría aún más a la realidad son los depredadores. En este caso el modelo de los peces tiene que ser modificado y tiene que contemplar la situación en la que pueden ser atacados. La energía puesta en juego en el movimiento, la edad o la muerte de los peces, no están contempladas actualmente en el modelo. La inclusión de estos parámetros incrementará aún más la realidad del sistema simulado.

Con el objetivo de incrementar la realidad del modelo y simular ecosistemas cada vez más complejos, es imperioso reducir los tiempos de simulación vinculados con las comunicaciones. Para tal fin una alternativa sería replantear el sistema de comunicaciones. Dicho cambio se basaría en evitar los mensajes de consulta de petición de información. Una solución propuesta es que luego de cada iteración se envíe a los LPs vecinos, la información de los individuos que se encuentran cerca de la frontera y que pueden afectar a los individuos de los LPs vecinos. De esta forma se enviaría un solo mensaje y se evitarían los tiempos de sincronización en el envío y recepción de mensajes.

A esta alternativa del envío de mensajes individuales se puede añadir una reducción en la complejidad básica del algoritmo, $O(N^2)$, donde N es la cantidad de peces en el LP. Si el espacio del LP es dividido en cubos con lados iguales al radio máximo de influencia (R_3), la consulta de los vecinos potenciales en vez de hacerse sobre todo el LP, se reducirá a la consulta de los individuos que se encuentran en los cubos adyacentes al que se encuentra el pez.

Las ideas utilizadas en este trabajo pueden ser llevadas a otros campos en los que se trabaje con modelos orientados al individuo. La metodología aplicada podría utilizarse para simular el comportamiento humano en situaciones límites, y poder predecir cómo será el flujo de movimiento de las personas. Esta herramienta podría ser utilizada, por ejemplo, para diseñar en forma adecuada la ubicación de las salidas de emergencia.

Bibliografía

[1] J. M. Alonso and R. B. Palacio, "Simulación Paralela de Sistemas de Sucesos Discretos" Disponible en: http://www.sc.ehu.es/acwmialj/documents/survey.ps 2004.

[2] P. L. Andrews, "BEHAVE: Fire Behavior prediction and modeling systems - Burn subsystem, part 1" General Technical Report INT-194. Odgen, UT, US Department of Agriculture, Forest Service, Intermountain Research Station 1986.

[3] P. L. Andrews, C. D. Bevins, and R. C. Seli, "BehavePlus fire modeling system, version 3.0: User's Guide" Gen. Tech. Rep. RMRS-GTR-106WWW Revised. Ogden, UT: Department of Agriculture, Forest Service, Rocky Mountain Research Station. 132p, 2005.

[4] I. Aoki, "A simulation study on the schooling mechanism in fish" *Bulletin of the Japan Society of Scientific Fisheries,* vol. 48, pp. 1081-1088, 1982.

[5] D. Ball and S. Hoyt, "The adaptive Time-Warp concurrency control algorithm" en *SCS Multiconference on Distributed Simulation* San Diego, Californa: Society for Computer Simulation. Simulation Series, Vol. 22, Num. 1, 1990, pp. 174-177.

[6] J. Banks, J. S. Carson, and B. L. Nelson, *Discrete-event system simulation*, 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, 1995.

[7] J. Bascompte, J. Flos, and R. Margalef, *Orde i caos en ecologia*: Publicacions Universitat de Barcelona, 1995.

[8] L. Berec, "Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis" *Ecological Modelling,* vol. 150, pp. 55-81, 2002.

[9] O. Berry, "Performance evaluation of the Time Warp distributed simulation mechanism" Ph.D thesis, Dept. of Computer Science: University of Southern California, 1986.

[10] G. H. Bham and R. F. Benekohal, "A high fidelity traffic simulation model based on cellular automata and car-following concepts" en *Transportation Research Part C: Emerging Technologies*. vol. 12, no1: Elsevier Science, 2004, pp. 1-32.

[11] G. Bianchini, A. Cortes, T. Margalef, and E. Luque, "Improved prediction methods for Wildfires using High Performance Computing: A comparison" en *ICCS 2006 - International Conference on Computational Science* University of Reading, UK: LNCS 3991, 2006, pp. 539-546.

[12] G. Bianchini, A. Cortés, T. Margalef, and E. Luque, "S2 F2 M - Statistical System for Forest Fire Management" en *ICCS 2005 - International Conference on Computational Science* Emory University Atlanta, USA: LNCS 3514, 2005, pp. 427-434.

[13] G. Bianchini, A. Cortés, T. Margalef, E. Luque, E. Chuvieco, and A. Camia, "Wildland Fire Propagation Danger Maps Based on Factorial Experimentation" en *Information Technologies in Environmental Engineering (ITEE'2005)*: Shaker Verlag, 2005, pp. 173-185.

[14] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*: Addison-Wesley Professional; 1st edition (September 30, 1998), 1998.

[15] R. E. Bryant, "Simulation of Packet Communications Architecture Computer Systems" MIT-LCS-TR-188, Massachusetts Institute of Technology, 1977.

[16] R. Buyya and C. Szyperski, *Cluster computing*. Huntington, New York: Nova Science Publishers, 2001.

[17] W. Cai and S. J. Turner, "An algorithm for distributed discrete-event simulation - the carrier null message" en *Proceeding of SCS Multiconference on Distributed Simulation* Vol. 22, No 1, 1990, pp. 3-8.

[18] F. Cores, "Switch Time Warp: Un método para el control del optimismo en el protocolo de simulacion distribuida Time Warp" en *MsC, DACSO - Departamento de Arquitectura de Computadores y Sistemas Operativos* Barcelona: UAB - Universidad Autónoma de Barcelona, 1999.

[19] I. D. Couzin, J. Krause, R. James, G. Ruxton, and N. Franks, "Collective Memory and Spatial Sorting in Animals Groups" *Journal of Theoretical Biology,* vol. 218, pp. 1-11, 2002.

[20] K. M. Chandy and J. Misra, "Distributed Simulation: A case Study in Design and Verification of Distributed Programs" *IEEE Transaction on Software Engineering,* vol. SE-5(5), pp. 440-452, 1979.

[21] D. L. Deangelis and L. J. Gross, *Individual-Based Models and Approaches in Ecology: Populations, Communities and Ecosystems*: Chapman & Hall, 1992.

[22] E. Deelman, T. Caraco, and B. K. Szymanski, "Parallel Discrete Event Simulation of Lyme Disease" en *Pacific Symposium* Hawaii: World Scientific Publishing Corp., 1996, pp. 191-202.

[23] S. M. Duke-Sylvester and L. Gross, "Integrating Spatial data into an Agent based Modelling system, ideas and lessons from the development of the Across Trophic Level System Simulation(ATLSS)" Integrating Geographic Information System and Agent-Based Modeling Techniques for Simulating Social and Ecological Processes, H.R. Gimblett, ed., Oxford Univ. Press.,2002, pp. 125-136.

[24] A. Ferscha, "Parallel and Distributed Simulation of Discrete Event Systems" en *Handbook of Parallel and Distributed Computing*: McGraw-Hill, 1995.

[25] M. A. Finney, "FARSITE: Fire Area Simulator-model development and evaluation" *Res. Pap. RMRS-RP-4, Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station,* p. 47, 1998.

[26] P. A. Fishwick, J. G. Sanderson, and W. F. Wolf, "A Multimodeling Basis for Across-Trophic-Level Ecosystem Modeling: The Florida Everglades Example" en *SCS Transactions on Simulation, Vol 15 (1)*, 1998.

[27] R. M. Fujimoto, "Parallel discrete event simulation" en *Communications of the ACM*. vol. 33(10), 1990, pp. 30-53.

[28] R. M. Fujimoto, "Performance of Time Warp under synthetic workloads" en *Proceeding of the SCS Multiconference on Distributed Simulation*. vol. 22(1), 1990, pp. 23-28.

[29] R. M. Fujimoto, "Time warp on a shared memory multiprocessor" *Trans. Soc. Comput. Simul. Int.,* vol. 6(3), pp. 211-239, 1990.

[30] A. Geist, *PVM : parallel virtual machine : a users' guide and tutorial for networked parallel computing*. Cambridge, Mass.: MIT Press, 1994.

[31] S. Habata, M. Yokokawa, and S. Kitawaki, "The earth simulator system" NEC Res. & Develop., 2003, pp. 21-26.

[32] R. Haberman, *Mathematical models: mechanical vibrations, population dynamics, and traffic flow : an introduction to applied mathematics*. Englewood Cliffs, N.J.: Prentice-Hall, 1977.

[33] M. Hanzich, F. Giné, P. Hernández, F. Solsona, and E. Luque, "Using On-The-Fly Simulation For Estimating the Turnaround Time on Non-Dedicated Clusters" en *EuroPar 2006, LNCS 4128, Vol 4128*, 2006, pp. 177-187.

[34] M. Hanzich, J. Lerida, F. Gine, F. Solsona, and P. Hernandez, "Using Simulation, Historical and Hybrid Estimation for Enhancing Job Scheduling on NOWs" en *IEEE International Conference on Cluster Computing*, 2006.

[35] M. Huston, D. DeAngelis, and W. Post, "New computer models unify ecological theory" *BioScience,* vol. 38, pp. 682-691, Nov 1988.

[36] A. Huth and C. Wissel, "The Simulation of fish schools in comparison with experimental data" *Ecological Modelling,* vol. 75, pp. 135-145, 1994.

[37] A. Huth and C. Wissel, "The Simulation of the movement of fish schools" *Journal of Theoretical Biology,* vol. 156(3), pp. 365-385, 1992.

[38] D. R. Jefferson and H. Sowizral, " Fast concurrent simulation using the Time Warp mechanism" en *SCS Distributed Simulation Conference*, San Diego, CA; (USA), 1985, pp. 63-69.

[39] O. P. Judson, " The rise of the individual-based model in ecology " *Trends in Ecology & Evolution,* vol. 9(1), pp. 9-14, 1994.

[40] J. Kreft, G. Booth, and J. Wimpenny, "BacSim, a simulator for individual-based modelling of bacterial colony growth" *Microbiology,* vol. 144, pp. 3275-3287, 1998.

[41] Y.-B. Lin and E. Lazowska, "Determining the global virtual time in a distributed simulation" Report. Dept. of Computer Science, University of Washington, Seattle, Washington 1989.

[42] H. Lorek and M. Sonnenschein, "Using parallel computers to simulate individual-oriented models in ecology: A case study" en *European Simulation Multiconference (ESM)*, 1995, pp. 526-531.

[43] J. H. Matis and T. Kiffe, *Stochastic population models: a compartmental perspective*. New York: Springer, 2000.

[44] F. Mattern, "Efficient Algorithms of Distributed Snapshots and Global Virtual Time Approximation" *Journal of Parallel and Distributed Computing,* vol. 18(4), pp. 423-434, August 1993.

[45] E. McCauley, W. G. Wilson, and A. M. d. Roos, "Dynamics of Age-Structured and Spatially Structured Predator-Prey Interactions: Individual-Based Models and Population-Level Formulations" *The American Naturalist,* vol. 142 (3), pp. 412-442, Septiembre 1993.

[46] J. G. Michopoulos and S. Lambrakos, "On the Fundamental Tautology of Validating Data-Driven Models and Simulations" en *ICCS 2005 - Computational Science -LNCS 3515* Atlanta, GA, USA: Springer Berlin / Heidelberg, 2005, pp. 738-745.

[47] J. Misra, "Distributed discrete-event simulation" *ACM Computing Surveys (CSUR),* vol. 18, pp. 39-65, 1986.

[48] D. Mostaccio, "Simulación Distribuida Aplicada a Modelos Orientados al Individuo Utilizando Algoritmos Conservadores" en *MsC, DACSO - Departamento de Arquitectura de Computadores y Sistemas Operativos* Barcelona: UAB - Universidad Autónoma de Barcelona, 2004.

[49] D. Mostaccio, R. Suppi, and E. Luque, "Distributed Events Simulation for Individual Oriented Models" en *EMSS 2005 - European Modeling Simulation Symposium*. vol. 1 Marsella - Francia: IM3 - International Mediterranean Multimodeling Multiconference, 2005, pp. 105-110.

[50] D. Mostaccio, R. Suppi, and E. Luque, " Distributed Simulation of Ecologic Systems" en *XVI Jornadas de Paralelismo*. vol. 1 Granada - España: Thomson, 2005, pp. 671-676.

[51] D. Mostaccio, R. Suppi, and E. Luque, " Distributed Simulation of Large-Scale Individual Oriented Models" en *Journal of Computer Science & Technology*. vol. 6 (2), 2006, pp. 59-65.

[52] D. Mostaccio, R. Suppi, and E. Luque, "Simulación Distribuida de Modelo Orientados al Individuo utilizando MPI" en *CACIC 2004 - X Congreso Argentino de Ciencia de la Computación*, La Matanza - Argentina, 2004.

[53] D. Mostaccio, R. Suppi, and E. Luque, "Simulation of Ecologic Systems Using MPI" en *EuroPVM/MPI 2005 - LNCS 3666*, Sorrento - Italia, 2005, pp. 449-456.

[54] P. Munt, "Simulació distribüida en PVM: implementaciò dels algorismes Time Warp i Switch Time Warp" en *Proyecto de Fin de Carrera, DACSO - Departamento de Arquitectura de Computadores y Sistemas Operativos*: UAB - Universidad Autónoma de Barcelona, 1999.

[55] P. S. Pacheco, *Parallel programming with MPI*. San Francisco, Calif.: Morgan Kaufmann, 1997.

[56] J. K. Parrish, S. Viscido, and D. Grünbaum, "Self-organized fish schools: An examination of emergent properties" *Biol. Bull,* vol. 202, pp. 296-305, 2002.

[57] B. R. Preiss and W. Louck, "Prediction and Lookahead in Distributed Simulation" Report. Computer Communications Group, University of Waterloo 1989.

[58] G. Proctor and C. Winter, "Information Flocking: Data Visualisation in Virtual Worlds Using Emergent Behaviours" en *Virtual Worlds 98 - LNCS*. vol. 1434: Springer Berlin / Heidelberg, 1998, pp. 168-176.

[59] M. Quinn, *Parallel programming in C with MPI and OpenMP*: McGraw-Hill, 2004.

[60] W. L. Romey, "Individual differences make a difference in the trajectories of simulated schools of fish" *Ecological Modelling,* vol. 92, pp. 65-77, 1996.

[61] B. Samadi, "Distributed simulation, algorithms and performance analysis" Ph.D thesis. University of California, Los Angeles, 1985.

[62] T. Sato, S. Kitawaki, and M. Yokokawa, "Earth Simulator Running" en *ISC2002 - International Supercomputing Conference*, Heidelberg, 2002.

[63] M. Serrano, R. Suppi, and E. Luque, "Parallel Discrete Event Simulation. State of the Art and Bibliography" Report. DACSO - Departamento de Arquitectura de Computadores y Sistemas Operativos. UAB - Universidad Autónoma de Barcelona, Barcelona 1998.

[64] M. D. F. Shirley, S. P. Rushtona, G. C. Smithb, A. B. Southa, and P. W. W. Lurza, "Investigating the spatial dynamics of bovine tuberculosis in badger populations: evaluating an individual-based simulation model" *Ecological Modelling,* vol. 167, pp. 139-157, 2003.

[65] J. M. Smith, *Models in ecology*. Cambridge [Eng.]: University Press, 1974.

[66] L. M. Sokol, D. P. Briscoe, and A. P. Wieland, "MTW: a strategy for scheduling discrete simulation events for concurrent execution" en *SCS Multiconference on Distributed Simulation* San Diego, CA (USA): Vol 19(3), 1988, pp. 34-44.

[67] T. Sterling, J. Salmon, D. Becker, and D. Savarese, *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*: The MIT Press, 1999.

[68] T. L. Sterling, *Beowulf cluster computing with Linux*. Cambridge, Mass: MIT Press, 2002.

[69] W. R. Stevens, B. Fenner, and A. M. Rudoff, *UNIX network programming*, 3rd ed. Boston, MA: Addison-Wesley, 2004.

[70] W. K. Su and C. L. Seitz, "Variants of the Chandy-Misra-Bryant distributed discrete-event simulation algorithm" en *SCS Multiconference on Distributed Simulation*. vol. 21(2), 1989, pp. 38-43.

[71] R. Suppi, D. Fernández, and E. Luque, "Fish Schools: PDES Simulation and Real Time 3D Animation" en *Fifth International Conference on Parallel Processing and Applied Mathematics - LNCS 3019*. vol. 3019: Springer Berlin / Heidelberg, 2004, pp. 505-512.

[72] R. Suppi, P. Munt, and E. Luque, "Using PDES to simulate Individual-Oriented Models in Ecology: A case study" en *ICCS 2002 - Proceedings of the international Conference on Computational Science-Part I - LNCS 2329* Amsterdam, The Netherlands: Springer-Verlag, 2002, pp. 107-116.

[73] R. A. Tapia, C. Lanius, C. M. M. Zeal, and T. A. Parks, "Computational Science: Tools for a Changing World. Rice University" Disponible en: http://ceee.rice.edu/Books/CS/index.html

[74] A. Verbraeck, Y. A. Saanen, and E. C. Valentin, "Logistic Modeling and Simulation of Automated Guided Vehicles" A. Bargiela and E. Kerckhoffs, Eds. Simulation Technology: Science and Art. 10th European Simulation Symposium and Exhibition, 1998, pp. 514-519.

[75] B. Wilkinson and C. M. Allen, *Parallel programming : techniques and applications using networked workstations and parallel computers*, 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.

[76] X. Y. Yang, P. Hernández, F. Cores, A. Ripoll, R. Suppi, and E. Luque, "Providing VCR in a Distributed Client Collaborative Multicast Video Delivery Scheme" en *Euro-Par 2006, LNCS 4128*, Dresden Germany, 2006.

[77] X. Y. Yang, P. Hernández, F. Cores, L. Souza, A. Ripoll, R. Suppi, and E. Luque, "DVoDP2P: Distributed p2p assisted multicast vod architecture" en *IEEE International Parallel & Distributed Processing Symposium (IPDPS'06)* Rhodes Island, Greece, 2006.

Referencias Web:

[78] Definition and Applications of Computational Science

http://www.aspire.cs.uah.edu/textbook/compsci_n1.html . Accedida en Enero 2007

[79] Overview of Computational Science

http://www.shodor.org/chemviz/overview/compsci.html . Accedida en Enero 2007